AD-A115 533 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
A DISTRIBUTED PROCESSING DESIGN FOR THE PERSONNEL DATA SYSTEM'S--ETC(U)
DEC 81 R V BRANDT
UNCLASSIFIED AFIT/GCS/EE/81D-4 NL

1 OF 2
AD
A115533

AD

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

82 06 14 201

AFIT/GCS/EE/81D-4

A DISTRIBUTED PROCESSING DESIGN
FOR THE PERSONNEL DATA SYSTEM'S
CENTRAL SITE

THESIS

AFIT/GCS/EE/81D-4 Ronald V. Brandt
Capt USAF

Approved for public release; distribution unlimited.

AFIT/GCS/EE/81D-4

A DISTRIBUTED PROCESSING DESIGN FOR THE PERSONNEL DATA SYSTEM'S CENTRAL SITE

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements of the Degree of

Master of Science

by

Ronald V. Brandt, B.B.A.

Capt USAF

Graduate Computer Science

December 1981

Approved for public release; distribution unlimited.

Preface

The purpose of this thesis was to analyze the central site portion of the Personnel Data System, and propose a distributed processing system design for implementation. A thorough review of applicable distributed systems theory was to be the basis for development of a design strategy which would employ appropriate concepts to improve system support for all users.

The design developed is based on analysis of system functions employed utilizing available terminals connected to the central site, and proposes distributed clusters of terminals developed through an allocation process based on precedence consideration of user location, system function usage similarity, and the office location of individual terminals. Several additional areas of future research were identified which will require expansion of current system documentation in order to employ some of the design techniques described.

I extend thanks to Col G. A. McCall, Director, Directorate of Personnel Data Systems, sponsor of this thesis effort and his entire staff for support in my data collection efforts. Specific appreciation is due MSgt (retired) Roger M. Davis for his diligent efforts in accumulating and dispatching the system reports which form the basis of the user database, and to my thesis advisor, Dr. Thomas C.

Hartrum, for his ability to provide direction when necessary, but allowing self-expression when appropriate. Additional thanks are due Dr. Gary B. Lamont and Maj Michael R. Varrieur for their support and guidance as committee members. Finally, I wish to express a measure of my gratitude to my wife, Judy, and son, Ronnie, for their ever-present assistance and understanding without which this thesis could not have been accomplished.

Ronald V. Brandt

Contents

Contents

List of Figures

Abstract

A distributed processing system design strategy was developed and applied to the central site portion of the Personnel Data System at the Air Force Manpower and Personnel Center (AFMPC), Randolph Air Force Base, Texas. A software package was developed to support building of a user database and the clustering of users based on geographic locations and similarity of system function use. Design considerations such as data redundancy and concurrency controls are discussed, along with the concept of Conflict Graph Analysis developed to support the System for Distributed Databases.

Software documentation and similarity tables depicting the distributed system design involving 425 terminals clustered into 46 user groupings are included. The design as presented requires further development to include data access requirements and definition of lower levels of system function utilization. Development of a simulation package to validate the final system design is recommended.

A DISTRIBUTED PROCESSING DESIGN FOR THE PERSONNEL DATA SYSTEM'S CENTRAL SITE

I Introduction

Background

The Personnel Data System (PDS) central site, located at the Air Force Manpower and Personnel Center (AFMPC), Randolph AFB, Texas, is the hub of the world-wide PDS which provides up-to-date personnel information to all major Air Force activities. The central site is the repository for personnel records on every member of the total force Air Force community (active duty military and civilian, Air National Guard, and the Air Force Reserve) and provides on-line file access through more than 500 terminals and 16 minicomputers, to personnel managers at the AFMPC, Pentagon, Air Reserve Personnel Center, Headquarters Air Force Reserve, Office of Civilian Personnel Operations, and most major commands and separate operating agencies.

The hardware environment comprising the current central site is described in Appendix A (Ref 1:3-6). All of this equipment will be replaced in the next several years as part of the AFMPC competitive reacquisition program (Ref 1). Much of the design strategy developed in this investigation is based on utilization of the equipment capabilities described for reacquisition, or projected capabilities after equipment upgrade.

Problem Statement

User demands for new processing power have greatly increased during the past twenty years requiring several system expansions and continual searches for processing optimization areas. Increased workloads have repeatedly caused system response time degradations and poor turnaround time for batch applications. Additionally, usage of many new applications, developed in response to perceived user needs, have had to be tightly controlled to prevent further adverse system impact.

Such current system limitations have generated considerable interest in identifying ways to improve response to all user needs. One such area of interest is the feasibility of applying distributed processing system concepts to the PDS central site. Successful implementations of a distributed system can offer significant benefits to users in terms of improved response times, availability, and flexibility/adaptability.

The purpose of this work then is to perform an analysis of the current system, propose a distribution strategy, and develop a distributed system design supporting central site operations and providing the benefits described above to all users.

Scope

Processing currently encompassed in the PDS central site (not ancillary processing at base level or on dedicated

equipment at other levels) will be the basis for analysis. The central site environment forming the basis for this analysis is portrayed in Figure 1-1. No effort will be made to perform a communication network analysis, or to require structural changes in the central site computer support structure. Distributed nodes will be formed using a first priority of location commonality to avoid increasing communication requirements. Software development will be limited to that required to support the distributed design analysis process.

Assumptions and Constraints

The design of a distributed system would typically include tailoring the attributes of each node to the requirements of the users to be supported. These attributes would include numbers of terminals, disk storage, tape drives, memory, and card readers/punches. Many of these attributes have been defined to some extent by the AFMPC reacquisition description of the remote batch terminals which will serve as the basis for the distributed system node computers. These attributes are included below as design constraints.

- Any system functions not included in the distributed environment are dependent on central site capabilities exclusively, or are part of the Microform system which is excluded from this analysis.

- The hardware attributes required for each distributed

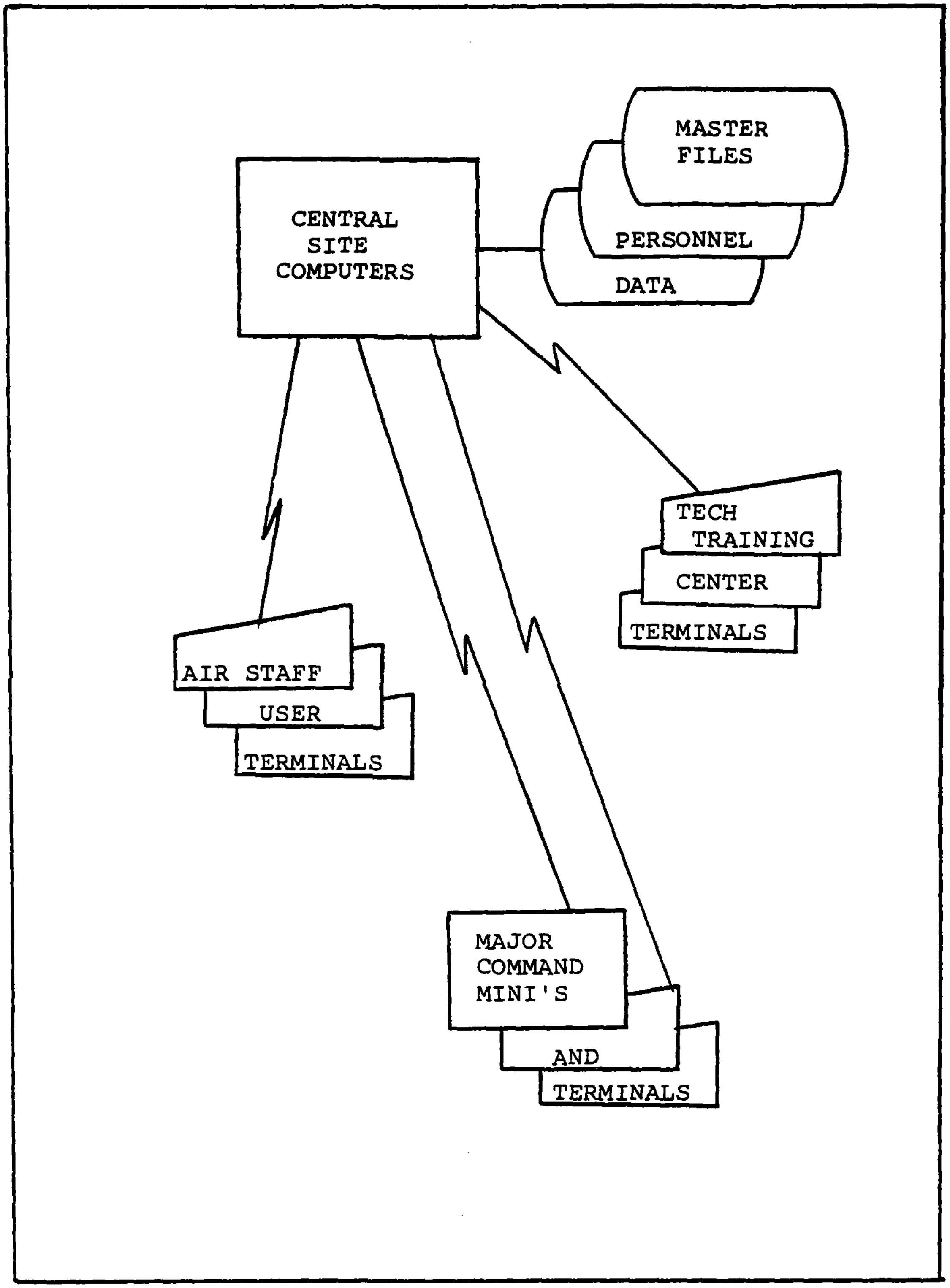

Fig 1-1. PDS Central Site System

system node must not exceed the projected capabilities of the remote batch terminal (RBT) in the AFMPC reacquisition program. These capabilities include: 100 MB intermediate storage, communications support for one or two 9600 BPS lines, operator console, 9-track 1600 BPI magnetic tape drive, card reader, and 10 KBPS communications capability for serving as a concentrator for 4 to 16 CRT devices. The reacquisition plan calls for a minimum order of twenty such devices, with possible growth to 53.

- Documentation of current system processing is available to the extent of correlating terminal identifiers, usercodes, system functions or groups of functions, and personnel transaction identifiers (PTI) input under the PERSTRANS system function. Additional detailed information cannot be obtained without implementing software monitors. Such software could not be developed and implemented in the time available, but will be discussed in Chapter V for recommended future actions.

- Documentation of user processing requirements supported by the Procurement Management System (PMS) is essentially non-existent. Additionally, the interactive structure of PMS, with on-line data change, demands considerable detailed analysis of system operations to develop a system strategy for data concurrency management and the prevention of deadlocks and lost updates as discussed later in Chapter II. Obtaining the necessary documentation would require manpower costs at AFMPC which cannot be justified at this

time, precluding the inclusion of the PMS in the distributed system design being developed.

- The lack of information linking each terminal user to the data items accessed through all system functions prevents the expansion of the similarity table and use of conflict graph analysis processes to further develop the detailed design.

General Approach

Three initial efforts were begun concurrently as the project was started. First was a review of available literature in the area of distributed systems design and implementation. The results of the review are documented in Chapter II, where the theoretical development is described. Secondly, several sources of information concerning current central site operations were identified and recent historical data was requested. This information and some insights into system evolution and future plans, based on past experience and continuing contact with system managers at AFMPC, contributed to the description developed in Chapter III. The third part of these initial efforts was the design and implementation of software which supports creation of the database used to build descriptions of system users. Documentation of this program is included as Appendix B, while the description of program use is included in Chapter IV.

Upon completion of the terminal use database, similarity

matrices were created which portrayed the degree of commonality among users in the same geographical locations. These user groupings, or clusters, were established through an iterative process using target minimal similarity values, and provided the basis for applying the constraints of the reacquisition hardware in arriving at the initial design. The database employed in generating these clusters was validated during a visit to AFMPC in August 1981, and was updated to a current position as of 1 July 1981.

II Theoretical Basis

Introduction

Distributed processing, distributed systems, dispersed systems, networked systems, and distributed databases are only a few of the terms which seek to define the concept of providing computing resources closer to the intended user community. Discussions of distributed processing concepts reflect the continuing confusion that exists because a common definition has not been found. One reason for the continuing discussion and a great deal of the confusion results from the fact that these terms, and many others (Ref 11:87-104), refer to specific applications which are somewhere along a continuum of application possibilities. This continuum results from the diverse needs of the thousands of organizations that possess computer processing capabilities that evolve as the organization evolves and as technological advances provide capabilities previously unavailable or too costly.

Distributed Processing System Structures

Two distributed system structures are common to most literature on this subject (Ref 4:39-43): horizontally and hierarchially distributed systems. Most authors are eager to avoid categories that are too narrow, and will also include a "composite system", sometimes referred to as a hybrid system (Ref 4:44).

Horizontally Distributed Systems. An example of a horizontally distributed system, depicted in Figure 2-1, consists of three processing centers which would probably have been distributed primarily on the basis of location or function. Such a structure would normally be created by interfacing three previously autonomous processing centers to increase the overall capabilities of the system by supporting resource sharing, data exchange, and workload leveling between system nodes. This type of distributed design can also provide the capability for a higher headquarters computer to extract data for consolidated reports listing information from all organization functions.

Hierarchially Distributed System. While horizontal distribution typically comes into being by interaction of previously autonomous processing centers, the hierarchial system, illustrated in Figure 2-2, results from the growth of a single central processing center system. The expansion of this centralized system might occur as the result of growth in the number of applications required by system users. Rather than adding increased processing power to the existing processing center, the creation of subordinate centers may be more attractive because of cost criteria, long range plans for further additions to the system, or the fact that the current system has reached the limits of its expansion capabilities. The distributed processing center created in this

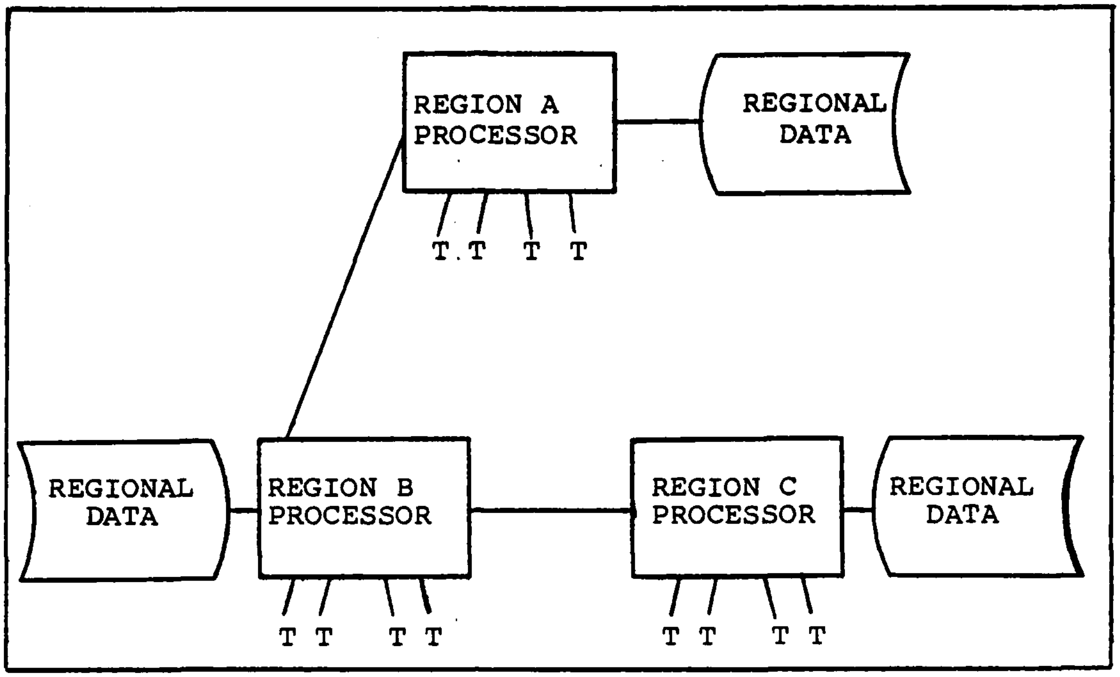

Fig 2-1. Horizontally Distributed System

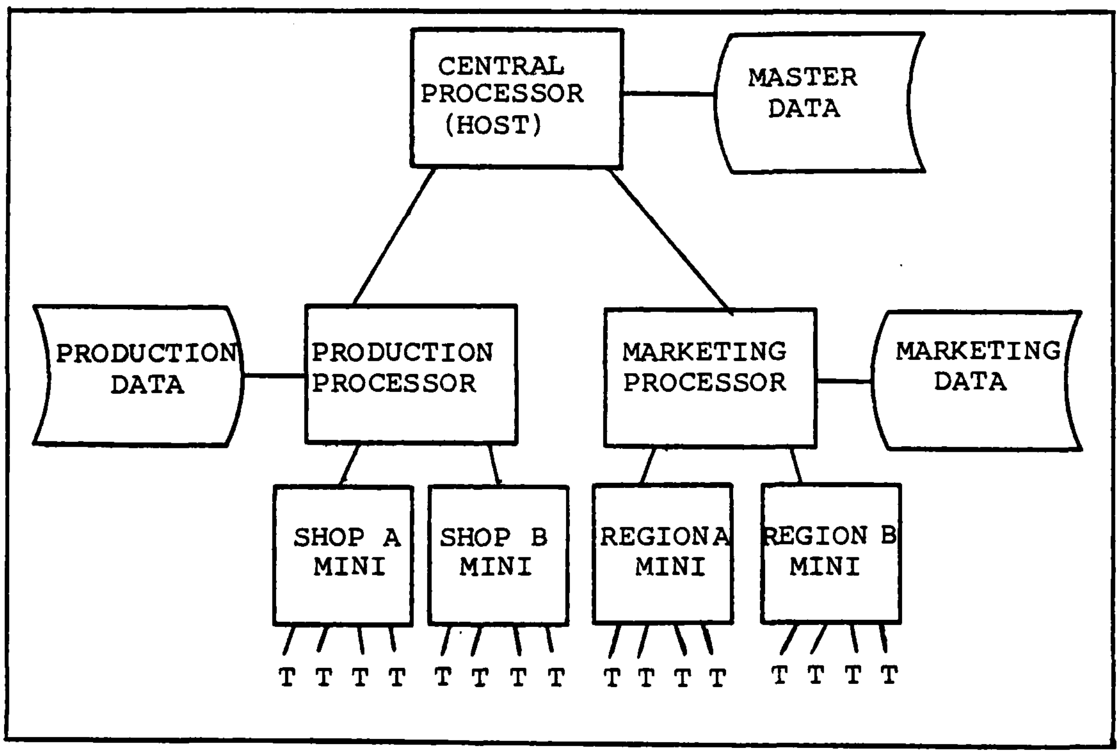

Fig 2-2. Hierarchial Distributed System

manner can be specialized in that it is designed to support one or more specific applications which are a small subset of applications supported by the central, or host computer.

Typically, a hierarchial distribution is based on the desire to distribute processing at the system level where it can be accomplished most effectively as measured by a cost/ performance ratio. Some of the performance criteria which may be important to this analysis are discussed later in this chapter.

Hybrid Distributed Systems. This category of distributed system structures is a kind of "catch-all" category for systems that do not neatly fit into either of the previously described categories. Most of the larger, more complex systems would fall into this category, because they include interconnection of generally autonomous processing centers, some having subordinate hierarchially distributed systems, and in some cases a headquarters host computer which extracts reporting information from all portions of the system for use by top level managers. As will be shown in Chapter III, the current Personnel Data System is a member of this hybrid category, and has been used as the model for creation of the example of the hybrid system provided in Figure 2-3.

Results of Distribution

Before discussing the possible techniques for developing a distributed system, the analyst must be aware of the

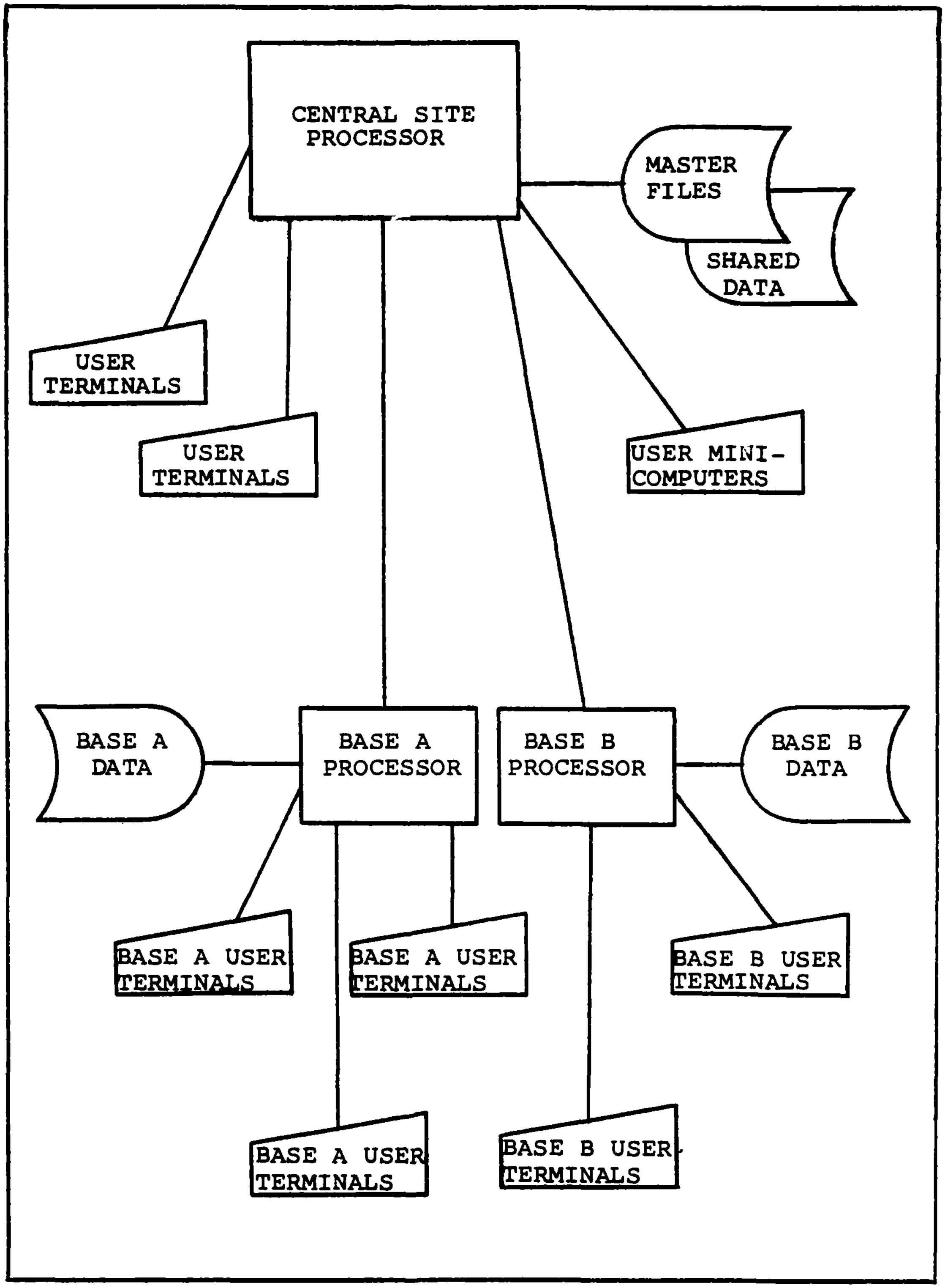

Fig 2-3. A Hybrid Distributed System

advantages and disadvantages of distribution, and develop an approach which meets the specific goals for the organization being supported. Several of the advantages have been touched upon in the preceding description of typical distributed system structures. The ability to design a system node which is specialized to perform a certain function will often achieve very attractive performance capabilities when compared to a multitalented node which must perform a diverse number of tasks equally well. Additionally, from a design viewpoint, as the complexity of a system increases the problems with maintenance of the system will also increase (Ref 20:18-21).

A second capability distributed systems provide is data exchange or sharing, in that data not available at a certain system node can be accessed over the system communications network. This is especially important in systems with many very large databases where too much data redundancy becomes unacceptably expensive. A user needing a single data item from a file can request that data for temporary use rather than being required to possess a local replication of the entire database. As with most system capabilities, the advantages normally have countervailing disadvantages which must be considered in the design analysis. As the number of requests for data from non-local databases increase, the communications traffic and cost will increase even though storage costs would be lower, because of fewer applications. One

optimization technique computes system costs for storage and transmission (Ref 7). The design can be structured to minimize costs once unit storage and transmission costs, file lengths, and request rates are available.

Load leveling is another benefit to be derived from a distributed system structure. Systems that evolve over time into some distributed mode often provide the capability to migrate applications or workload from one system node to another to relieve overloads or imbalances that develop. Offsetting this advantage is the possibility of increased system costs when peripherals or added resources must be placed at a system node to support possible workload migrations.

Another major advantage is the ability of a user at a distributed node to utilize some system processing capability not available at that user's local node. In a properly implemented system this capability would be transparent to the user who has no need to be concerned with where the actual process is performed. This resource sharing capability is faced with the same balancing of costs versus benefits described for data sharing, since the request and possibly data must be transferred between nodes to complete the action.

Other issues of concern in the design that may be important in meeting organizational goals are the ability to more easily restrict access to data in a distributed system simply because interconnection capabilities may be more easily controlled. Also, the possibility of improving

response time for critical applications can be achieved on the basis of nodal specialization, while system availability is enhanced by allowing users to be switched to other nodes or the central system when the local node fails.

Distribution of Users. A critical problem in developing a distributed system design is the determination of who is supported by which system node. Strict distribution patterns based on functional affiliations is a common approach which corresponds to the previously described horizontal distribution structure. The major adverse aspect of such a design would appear to be the system vulnerability to hardware failures because applications would most likely be available at only a single system node. This problem can be alleviated if the system also incorporates the hierarchial property where a node will possess a subset of the capabilities available at the host node. This enables support at the host or other nodes at the same horizontal system level, if the local node becomes unavailable.

Distributions based on database requirements can also encounter the insufficient redundancy problem unless some amount of data replication is achieved across the system. The amount of data redundancy required to achieve a desired level of availability is easily derived based on the probability of the local copy being inaccessible, multiplied by the probability of all other possible sources being inaccessable. In the fully connected distributed network, where

every node is connected to all other nodes in the system (Ref 4:65-66), the probability of being unable to access data is essentially eliminated since alternate access paths are established automatically when the primary data source or path is unavailable. Such availability enhancement, at the cost of communications lines between all nodes, is most beneficial in system where real-time data changes and access are essential to the achievement of organizational goals.

System Analysis and Design

Perhaps the best definition of the activities encompassed by the terms systems analysis and design is the identification and gathering of information about a problem, and the use of that information to formulate and evaluate potential solutions to those problems (Ref 19:18-19). Palmer (Ref 13) describes a structured top-down approach to designing distributed computing systems by developing "baseline" designs at four levels of system analysis. The four levels: subsystem/network, nodal, computer system, and hardware/software levels, allow early analysis to focus on assimilation and refinement of requirements which support design development for that system level, but more importantly provides the basis for carrying the design to the next level of analysis. Each baseline design is developed using a four-step sequence of activities: analysis, partitioning, allocation, and synthesis. Each of these activities is described in some detail below for the subsystem/network

level.

Analysis. This activity involves the identification and accumulation of user requirement information which details the functional and data entities, and interrelationships which exist in the system. The output from this activity would normally include a set of interrelated functions and data entities, and information describing the functional performance requirements (data loading, reliability, etc).

Partitioning and Allocation. Using the analysis information, the partitioning and allocation activities involve applying selected distribution criteria to develop candidate partitions, or groups/clusters of users. These clusters based on the distribution criteria would reflect the goals of the following allocation activity and "good" design practices. The partitioning criteria typically involve data sharing, processing, and precedence relationships which exist between system entities. Although statistical analysis and simulations may be employed to determine the relative values of candidate partitions, the numbers and types of nodes are more often determined to a significant extent by physical constraints such as the number and location of users to be supported (Ref 13:24-25).

Synthesis. The goal of the synthesis activity is to identify the interface and control requirements to maintain control and data linkages that may have become broken or more complex as a result of allocation activities. Concerns such

as data management to maintain multiple copies of data and maintenance of communications with respect to shared data are important activities that must be accomplished and verified.

Cluster Development Techniques

The development of clusters of users which will be supported by a single node in the distributed system can be accomplished in several ways. The method selected often depends on the constraints on system design resulting from cost and performance goals. There are also constraints based on equipment capabilities and even the organization's political environment which often must be taken into account in the development process.

Based on perfect knowledge of system functions, users, and cost/benefit tradeoffs in the communications area, it is theoretically possible to develop simulation programs which will forecast performance characteristics under the varying clustering concepts devised. In small systems it could be possible to apply graph theory to identify a minimum spanning tree of user nodes which minimize costs and maximize benefits to obtain an optimum clustering strategy. As system size increases, such capabilities become np-complete, requiring the use of some heuristic technique (Ref 13:25).

One technique which provides a basis for determining system distribution patterns is called "fuzzy clustering", and is based on creation of proximity matrices which show

the degree of commonality that exists between system entities. Developed by Buckles and Hardin (Ref 6), this technique can be employed at several levels of detail to present a fuzzy picture of existing relationships in the system being analyzed. A subset of the proximity matrix approach is required in some cases where the full complement of descriptive information is not available. This less expansive approach uses similarity matrices which present the percentage of system work that system entities have in common. The development of the similarity matrix is accomplished by dividing the number of common system actions that two users employ by the total number of system actions they used. This measure shows the amount of cohesion that exists between the users, and when applied across a system will present a measure of cohesion existing among users supported at a single distributed system node.

Figure 2-5 is the similarity matrix for five terminals located at the AFMPC (G62, G37, G20, G100, and B2 are the terminal identifiers). Based on comparison of the different system functions employed during the months of February and July 1981 (Fig 2-4), users of terminal G62 used six different system functions. During the same period, terminals G37 and G100 were used to access five system functions, all of which were used by G62 users. This usage information results in the computation of a similarity value of 83 percent (six different system functions employed, with five being commonly used at the terminals being analyzed) reflected in the sample

Terminal Id	System Functions
B2	PERST, CANDE, RUN, SURF, GURU, ATLAS, AIRS
G20	PERST, CANDE, RUN, SURF, GURU, ATLAS, AIRS
G37	CANDE, RUN, SURF, GURU, AIRS
G62	PERST, CANDE, RUN, SURF, GURU, AIRS
G100	CANDE, RUN, SURF, GURU, AIRS

Fig 2-4. Function Usage Example

Terminal Id	B2	G20	G37	G62	G100
B2	--	1.0	.71	.86	.71
G20	1.0	---	.71	.86	.71
G37	.71	.71	---	.83	1.0
G62	.86	.86	.83	---	.83
G100	.71	.71	1.0	.83	---

Fig 2-5. Example Similarity Matrix

similarity matrix. When comparing G62 usage with the data gathered for terminals G20 and B2, the resulting similarity value is 86 percent (seven function employed totally, with six being common). Carrying this process across the relationships of each of the user pairs results in the completed matrix developed in Figure 2-5. From a design viewpoint, if these five users were allocated to a single distributed node, the result would be a node requiring a total of seven system functions. With added definition of the files utilized and eventually the data items and transactions employed, each node design can be based on providing a minimal set of attributes which will achieve maximum support of the allocated user's requirements.

Data Distribution and Concurrency Control

Most of the previous discussion of distributed system structures has been based on differences in the evolution of their development and decisions concerning which system functions should be distributed to best support user requirements. These concerns were most closely related to the distribution of system intelligence and the need to migrate or develop software to support the distributed processing. Closely tied to this migration of intelligence is the need to develop a strategy for the distribution of data to support the distributed functions. Just as distributed system structures occur in several ways, database distribution is multifaceted and

involves design considerations which are vital to achievement of the system performance goals. Although complete development of a database distribution strategy was not possible in this thesis because data item usage could not be correlated to individual users, one data distribution strategy is described below and included in the recommendations portion of Chapter V.

Many of the advantages claimed for distributed processing systems, such as insulation from central site failures and decreased communications costs (Ref 4:29-34) are fully dependent upon continual access to system data. To insure this data access, the first step would appear to be the location of database copies at each of the distributed system nodes. If this approach were coupled with the fully connected network concept described earlier, data access problems can be virtually eliminated. Of course, such a "perfect system" approach cannot be economically justified in most cases. Thus, the better approach is to design the system within reasonable cost parameters that will approach the achievement of this perfection at a much lower cost.

Data distribution strategies cover a broad range of possibilities, from a fully redundant copy at every system node, to a single copy for the entire system (whether in one location or partitioned to various nodes). In centralized systems where many users may be accessing the same piece of

data at the same time, the primary concerns are the prevention of deadlock situations and "lost updates" (Ref 5:405). These problems are also of concern in distributed systems in addition to new problems of data concurrency among replicated copies of the same data items, and system operations when one or more system nodes fail.

Two common approaches to the prevention of deadlock and the lost update problems are enforcement of locking protocols or allowing deadlocks to occur and then "backing out" one of the involved transactions so the other may complete. Numerous locking approaches are documented in the literature along with proofs of their validity (Ref 18:324-356; 16). In complex system where multiple copies of data items may be maintained, use of these locking approaches results in serious service delays because system-wide locks are employed and must be synchronized. This synchronization can be accomplished by the utilization of system clocks (Ref 11:225) which enable the generation of individually unique "timestamps" for each system action, and the use of some PAR (positive acknowledgement and response) protocol.

Although such approaches will achieve data consistency eventually, applying data locking procedures across the system is cost prohibitive in distributed systems where multiple copies of data items are maintained (Ref 15:354-355). Recent work, especially for the System for Distributed Databases

(SDD-1) (Ref 16), presents the concept of conflict graph analysis that can be utilized during system design as a method for reducing the amount of concurrency control overhead.

In most current systems, data locking is employed to prevent the problems of lost update and deadly embrace discussed earlier. Once a system action has begun, the data to be used is locked, preventing access by any other user until the first action is completed. In a centralized system where internal communications exist at machine speed, such procedures are adequate. However a distributed system with data replications in several dispersed locations would soon cease to operate if all copies of the data had to be locked before an action could be started, and unlocked before the next could begin. The basic concept of conflict graph analysis is based on the fact that only a small proportion of the transactions will actually conflict with each other in a dangerous manner. By analyzing system transactions and assigning them to transaction classes which are then graphed to portray existing conflicts, the designer is able to develop a series of concurrency control actions, most of which will be considerably less complex and time consuming than system-wide locking (Ref 12:289-290).

The basis for data locking procedures as well as procedures employed in SDD-1 is the concept of serializability. The term serializability comes from the need to demonstrate that the result of two schedules for the accomplishment of

actions in two transactions are identical when one schedule involves interleaving of actions while the other schedule achieve the same result from serial completion of the operations. Figure 2-6 uses the transactions T1 and T2 to show that depending on the sequence in which two transactions are completed, the resulting impact on the database can be changed entirely. Without some background information indicating the intent of the users, it is impossible for any logic within the system to distinguish between the two "valid" results, 123-4567 and 124-5678. Thus, for this class of transactions, either result must be accepted by system designers as a valid result, and any schedule which obtains one of these results must be considered valid also. In SDD-1, concurrency control is based on the ability to obtain serializability in many cases without requiring data locking on all copies of a data item across the entire distributed system. In the case of transactions T1 and T2, as long as the transaction writes are completed in the same sequence in every case, the result will be consistent across the system. This consistent accomplishment can be assured by the implementation of a protocol requiring all system nodes to process such transactions in time-stamp order. This protocol will not satisfy all transactions; however, many transactions will satisfy the serializability concept with this protocol, or another that is less restrictive than one requiring transmission and acknowledgement of system-wide data locks/unlocks.

```
TRANSACTION T1:

      Replace PHONE-NR <1234567> Where NAME = 'ADAMS'.

TRANSACTION T2:

      Replace PHONE-NR <1245678> Where NAME = 'ADAMS'.

          SCHEDULE 1                    SCHEDULE 2

              T1                            T2
              T2                            T1

Result:  1245678                       1234567
```

Fig 2-6. Serial Transaction Schedules

Documentation of SDD-1 protocols which ensure serializability (Ref 3) include complete coverage of the concurrency control problem and prove the applicability of the conflict graph analysis approach to the other instances of dangerous transaction conflict. As will be discussed in Chapter V, documentation of PDS transactions, and analysis of distributed system nodes which are sources for input of these transactions would allow application of conflict graph analysis, and the capability for expansion of on-line update capabilities throughout the system while avoiding the overhead currently envisioned.

Summary

The theoretical foundation for the distributed processing system designed was developed from an analysis of the

literature on the subject and background knowledge of the central site environment. The current system is an example of a hybrid distributed system with elements of both the hierarchial and horizontal approaches. The general evolutionary pattern of the system reinforces this structure, and the central site distribution developed in this work will continue that evolution.

Two major benefits of distributed processing which have been discussed in this chapter appear extremely applicable to the solution of current PDS central site problems. As with most large systems, the continual requirement to develop and implement new and more powerful applications has tasked the growth potential of the currently available hardware. Expansion of current system capabilities, before the installation of the reacquisition hardware, will result in service degradations for many current users. Migration of functions to some distributed system nodes could immediately release central site resources for the required new capabilities, or to improve current system performance. Additionally, the adverse impact of hardware failures which occur more frequently with the older hardware which is currently used, can be avoided as a result of the "insulation" which is possible under distributed system concepts.

III PDS Description: Data Sources and Uses

Introduction

The Personnel Data System, as it exists today, is the result of a twenty year evolution caused by changing Air Force needs in the "people resource management" environment. As laws, policies, and social pressures have required Air Force managers to function differently, the PDS has been modified and expanded to better support these new functional requirements.

While the base level portion of the PDS continues to support the daily operations of base level manpower and personnel offices, the central site system has grown by incorporating increased amounts of data, new processing functions, and assuming workloads from other associated systems. Since the world-wide implementation of the Advanced Personnel Data System (APDS) in 1974, the central site has been the center of efforts to provide a fully integrated support environment for total force management in the Air Force. The current system structure was shown in Figure 1-1.

PDS Information

The core of the PDS central site system consists of the master data files which contain digital records for every individual in the total force Air Force, retired personnel, and individuals actively interested in being recruited into

the Air Force. These files, consisting of more than one million records, are maintained in an up-to-date condition primarily through batch update actions based on data flow from the base and Headquarters Air Force levels of the PDS.

These digital records are all similar in many respects, since they depict historical data about personnel actions relating to an individual. But there are unique aspects depending on the Air Force component with which the individual is associated. Typical information that is retained describes duty history, promotion history, training, information concerning current assignment and place of residence, and many other data items used by Manpower and Personnel managers when performing some action or making a decision affecting the individual.

Typically, a data item relates to one specific functional area such as promotions, assignments, or training, and an office within this functional area assumes management responsibility for that data item. These offices of primary responsibility (OPR's) in many cases are the only source of data changes to the items, or they are responsible for insuring changes made to the items by other system users are appropriate. Because of the size of the system the validations necessary to insure this data management are accomplished within the system. The necessary controls are documented by the OPR and implemented by the personnel of the Directorate of Personnel Data Systems (AFMPC/MPCD).

System Users

Both the variety and number of system users supported from the central site have increased rapidly during the last seven years. More than five hundred system terminals and 16 minicomputers currently provide access to system data for users world-wide. These users are at every level of Air Force Personnel and Manpower management, and are employed in every facet of these functional areas. The world-wide dispersion of users has resulted in the development of a large communications network to provide direct system access to the PDS information for these users. To present a clearer picture of the PDS user community, five major segments are described below.

San Antonio, Texas. More than 200 PDS terminals are located in this area, with the large majority being employed at the AFMPC, the Office of Civilian Personnel Operations (OCPO), and Headquarters, Air Training Command (ATC) at Randolph AFB, Texas. Other users groups are at Lackland, Kelly, and Brooks Air Force Bases.

Washington D.C. Air staff functions at the Pentagon, major commands at Langley and Andrews Air Force Bases, separate operating agencies (SOA's) at several locations, along with other Air Force and non-Air Force organizations are included in this widely diverse group of users in the Washington area.

Major Commands. Each of the Air Force Majcoms are supported to some extent by the PDS. User groups in this category that have not been mentioned previously are Pacific Air Forces (PACAF), United States Air Forces in Europe (USAFE), Alaskan Air Command (AAC), Strategic Air Command (SAC), Military Airlift Command (MAC), Air Force Communications Command (AFCC), and Air Force Logistics Command (AFLC). Each of these organizations utilize PDS information in the management of their personnel resources. The number of terminals at each location varies according to staff size and the volume of PDS workload represented.

Separate Operating Agencies and Direct Reporting Units. Users in these categories are widely dispersed across the continental United States. In most cases one or two system terminals have been provided to support the personnel management activities of the organization's staff. Included in this category are the Air Reserve Personnel Center (ARPC) and Headquarters, Air Force Reserves (AFRES) both of which have several system terminals because of the size of their staff and scope of responsibilities supported.

Technical Training Centers (TTCs). The TTCs at Chanute, Keesler, Lowry, and Sheppard Air Force Bases have recently been included in the PDS system access to support the training management requirements these users have that relate to system capabilities.

System Functions

As mentioned previously, the bulk of data maintenance accomplished in the system occurs as a result of batch update processes. Day-to-day interaction with the system for most users involves the need to determine the status of a person's record in one specific functional aspect, or having a change of status action which must be projected to occur at some future date. Many personnel actions recorded in the system are of the type where the action is projected to occur at some future date and the batch update of the system on the projected date will cause an automatic generation of appropriate transactions notifying interested personnel that the action has occurred. Printed notices are the typical method employed to document completed actions, with the appropriate office maintaining a copy of the notice for some period. Some functional offices supported by the system manage very large personnel programs such as airmen promotions and assignments, which preclude manual entry of all data changes. In such cases, one batch processing action will create output files containing the appropriate update transactions, and these files will be used as input files to the next update of the proper master file.

To support these kinds of user activities, the PDS provides a large number of system functions. There are probably more than sixty individual functions available on the

system; however, the ones described below are the most heavily used and of interest for the system design being developed.

SURF. The SURF function is a single record query capability. SURF is heavily used by almost all system users to access preformatted displays of portions of an individual's computerized personnel record. Access to the SURF function requires input of an approved usercode/password combination and selection of the SURF function for use on one of the master files. Once access to SURF and the file are established, the user must input the Social Security Account Number (SSAN) for the individual's record and the format number or individual data item number to be displayed. SURF displays typically provide information related to one personnel functional area of interest, with many master files having twenty or more standard display formats.

PERSTRANS. The PERSTRANS function, abbreviated "PERST" on reports, provides the capability for a user to enter a transaction into the system to update a single personnel record. These transactions may cause data changes, create a printed report, or project a future data change for the specified record. Access to this function is very similar to the requirements for the SURF function, with the transaction identification code being input to specify the appropriate file and transaction format. Although not as widely used as the SURF function, more than fifty percent of the users

studied in this thesis utilize the PERSTRANS function.

ATLAS. The ATLAS function supports the creation of batch queries against PDS files. Any file described in a file descriptor table can be accessed through ATLAS inquiries. These tables describe all of the attributes of the records in the file to be accessed. ATLAS queries in most cases will produce either a printed listing reflecting information requested by the user, or an output data file which can be used for batch update, generation of reports, further queries, etc. The majority of system users studied in this thesis have access to the ATLAS function.

Airmen Information Retrieval System (AIRS). The AIRS function is utilized by a limited group of system users to access data files which reflect airmen resource utilization throughout the Air Force.

GURU. The GURU function is a generalized file manipulation utility which allows users to create data and text files, review and alter print files before printing, and other data editing types of activities. Access to this function is widespread amongst users, serving as a replacement for the CANDE function described below.

CANDE. CANDE is a Burroughs Corporation system utility that performs many of the same functions as GURU. Since implementation of the GURU function about three years ago, use of CANDE has decreased considerably.

INFO. The INFO function is another older function that has been largely replaced by the GURU function. INFO's primary purpose is the creation of informational text files by various users, that could then be accessed by users across the system. Such files often contained schedules or system status notices which were maintained by system management personnel.

Personnel Data System Training (PDST). The PDST function is available to a limited number of system users that are involved in the management of Air Force training programs. This function includes the capability for real-time allocation of training quotas, and access to training programs status information.

PROMIS. The PROMIS system is also referred to as the Pipeline Management System (PMS) which supports the management of various aspects of the training pipeline. PROMIS users for the most part are the Armed Forces Entrance and Examining Stations (AFEES) throughout the United States where Air Force recruiters attempt to obtain entry authorizations for prospective Air Force recruits. A major part of the PROMIS support is the real-time person-job-match process which is used to identify Air Force requirements for which a specific applicant may qualify. The system runs under the control of the Burroughs database management system software. The size of the software supporting the PROMIS function necessitated

exclusion of PROMIS users from the distributed system design.

Other Functions. As indicated earlier, many other system functions are available to support PDS user requirements. Separate functions which support software testing, computation of retirement pay estimates, analysis of officer manning, and modelling of force structure changes are utilized by small groups of users according to their individual functional requirements. Unfortunately, system workload in these functions is not measured separately in any system reports and could not be reflected in the database built in this thesis.

Documentation of Function Usage

As discussed in Chapter II, the basis for user clustering was the similarity matrix which depicts the percentage of system function and files usage users have in common. The system documentation which provided the picture of user workload patterns was drawn from various system generated reports. Because the central site is comprised of multiple subsystems, no consolidated reports documenting system workload is produced. Therefore, the amount of information that was available to reflect user employment of system functions varied greatly. The various sources are described below, with a brief discussion included to indicate the limitations of some of the sources.

TERMSTRIP Reports. The "TERMSTRIP" reports provide a detailed picture of the transaction processing requirements

of each user and terminal using the PERSTRANS function. These reports include a list of all terminals which were the source of transactions for the period of the report, and the number of transactions input. In subsequent sections the terminal identifier, user identifier, time of transaction input, and the transaction PTI are recorded. The reports are generated twice daily as the input transactions are "stripped" to input files for later entry in batch updates. The average daily volume of transactions input under the PERST function exceeds five thousand, therefore a limited sample of these reports (twenty days) were used to identify the master files each terminal was used to access. As will be discussed in Chapter V, automation of the analysis of these reports would provide a valuable source of data usage information upon which to base expansion of the similarity/ proximity matrix analysis in future efforts.

TERMUSE Reports. These monthly reports describe the use of system functions by every terminal in the system. The information provided includes the terminal identifier, function usage in hours, number of transactions, and number of transactions per hour for 13 specific system functions and a grouping under the title "RUN PROG." Section two of this report lists the usercodes employed at each terminal during the period of the report. The information from this report was critical to the clustering process, in that it provided the basis for establishing a terminal usage pattern

which strongly suggested optimum assignment of terminals to distributed nodes based on the realities of actual system workload.

Other Sources. Additional system documentation came from reports listing all assigned usercodes including the user's location and office symbol, and a listing which described every system function and the size of the software supporting it, and the summary reject trend report which provided statistics by user identifier on the individual transactions input during the month of February 1981, and the percent of rejects found during the subsequent batch update process. Documentation of system workload in several areas are not created at any comparable level to these reports for the PERSTRANS function. Additionally, the grouping of as many as forty functions under the "RUN PROG" category prevents further definition of user workload for this investigation beyond the level of the TERMUSE report.

IV Distributed Design Development

Introduction

The evolution of the design strategy employed to arrive at the distributed system structure recommended for the PDS central site was perhaps more a result of the current system evolution than an embrace of theoretical concepts. The geographic dispersion of the PDS user community in many instances dictated establishment of a system node at a certain location although no strong system function cohesion existed. However, the four phase design approach presented by Palmer (Ref 12), and discussed in Chapter II was employed to obtain a structured methodology.

The intent of Chapter III was to present a clear depiction of current system operations and the user community being served, and to identify the sources and limitation of available system reports that reflected user requirements. These requirements, or system functions actually employed, could be utilized as one of the important attributes for developing the distributed design, as well as a measure of cluster cohesion reflecting the strength of the design. Having identified a structured approach for the design process and the information to be employed, the bulk of the design effort was ready to proceed.

User Database Development

Since several sources of information were available to document the system work accomplished by users, some linking workload to individual usercodes and others showing terminal identifiers, a common basis for analysis was required. The basis selected was the terminal identifier primarily as a recognition of the fluidity of the users within office areas. Because the present system authorizes system functions to usercodes, with essentially no restrictions based on which terminals may be used, users often "shop around" for the nearest available terminal. A review of the TERMSTRIP report documents this user movement, showing transactions input by the same user at several different terminals on the same day, and the use of one terminal by three or four users during a one hour period. Since the terminal identifier is the primary subject of the TERMUSE report containing the functions employed and the TERMSTRIP provides both the terminal and user identifiers, use of the terminal identifier would ensure the user fluidity was included in the database reflecting system operations.

Other important information required for depiction of system usage and for maintaining a record of user attributes were identified as the various data sources were reviewed. The final form of the user/terminal information record included the terminal identifier, office symbol or geographic location, and the list of system functions and master files

used at the terminal.

The user database was created using the software program described in the next section of this chapter. User records were built based on review of the summary reports of the TERMUSE report for March 1981 and were later updated to a condition as of 1 July 1981, after validation of the original database at AFMPC in August. System function usage information was added to each user record from the TERMUSE report, while master file requirements were added based on PERSTRANS transactions reflected in the TERMSTRIP reports.

The final version of the database contained 425 terminal records and more than 6000 system functions and master file designations. The validation visit to AFMPC primarily served to identify portable terminals and output-only devices such as printers and punches which were removed from the database.

Software Development

The iterative nature of the clustering process and the large number of users and system functions that would be included in the database strongly suggested the requirement for development of a set of automated tools. The requirements to be satisfied by these software tools included the ability to create terminal records containing the attributes described in the previous section, support of database maintenance, and the various types of clustering concepts to be employed during the design development process. Each of the specific

capabilities supported by the software is documented in the program listing provided as Appendix B. The use of the various clustering capabilities are described later in this chapter as the design process is described.

Partitioning and Allocation

The partitioning and allocation activities are discussed together for the subsystem/network level as a recognition of the impact of resource constraints on the partitioning activity at this level (Ref 13:23-25). The two overriding constraints which limited freedom in this process were the number and location of system nodes and the computer power available at the nodes. Descriptive information obtained from sources at AFMPC which described the purpose of each system function and the amount of memory required for the supporting software strongly influenced the final form of the user database described earlier. The size (76,000 lines of source code) and structure of the PROMIS subsystem dictated exclusion of users primarily using this function. Some users with small amounts of recorded PROMIS workload were retained if they employed the other system functions. The exclusion of these users is not meant to rule out the possibility of applying distributed system concepts to them in the future, but rather recognizes the fact that a considerably expanded analysis of this subsystem is required before inclusion in the distributed system could be evaluated. The elimination of this group of users

established the final form of the user database with 425 terminals as mentioned previously.

Criteria Selection and Thresholds. The criteria which are used to judge processing entities of the proposed system must be measureable and reflect the goals of the following design activities. For the PDS, as for any system with a wide dispersion of users, user location commonality must be a primary criterion. In several instances, reflected in Figure 4-1, groups of terminals can only logically be allocated to the same distributed node because of geographic location. Where large groups of terminals were located in the same area the other criteria would be applied to establish nodes of acceptable size.

After application of the locality criterion, the second criterion selected was the similarity of system functions employed at the terminals in the area. The goal of the allocation activity in applying this criterion would be to obtain the highest degree of cohesion for groups of terminals allocated to each distributed node. Ancillary to the similarity criterion was the establishment of similarity value thresholds for the clustering process and identification of the appropriate number of terminals to allocate to each node computer. As described by a AFMPC reacquisition project officer (Ref 5), the computers envisioned for use as distributed system nodes would be capable of supporting up to sixteen terminals each. If the system were designed with each node supporting a full

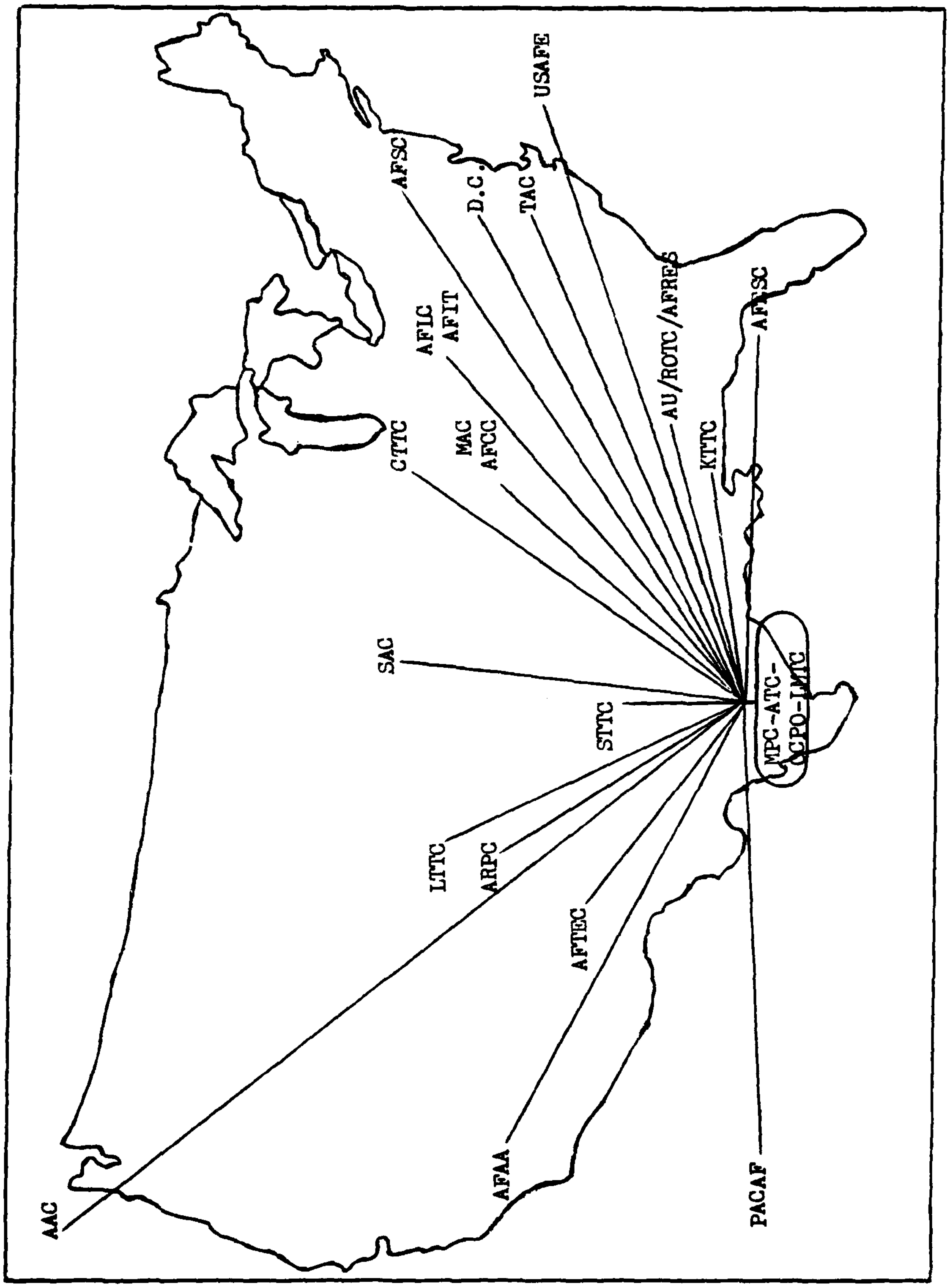

Fig 4-1. PDS Terminal Locations

complement of sixteen terminals, any new workloads or users would immediately necessitate reconfigurations. Therefore, a target of no more than twelve terminals at each node was selected. This means more nodes will be required to support the terminals included in the distributed plan, but each node will be somewhat less expensive, while considerable upward flexibility for future system changes will be provided.

Additional partitioning and allocation criteria which may be applied in the future, once additional information is available, are described in Chapter V.

Relationship Evaluation. The data flow diagram in Figure 4-2, depicts the transformation of the user database into the final distributed design consisting of 46 clusters. The number of clusters or nodes created by each transformation process are included in parentheses on the data streams arriving at the final transform, "Design Synthesis." Each of the transformations utilized one or more of the capabilities provided in the software developed as part of this thesis (Appendix B).

The application of the terminal location criterion identified 12 terminal clusters of acceptable size (not more than 12 terminals), 31 terminals scattered to various areas that could not be considered common to any of the other clusters, and eight clusters with more than twelve terminals each. Three of these initial clusters represented AFMPC, ATC, and OCPO located at Randolph AFB. Their transformation

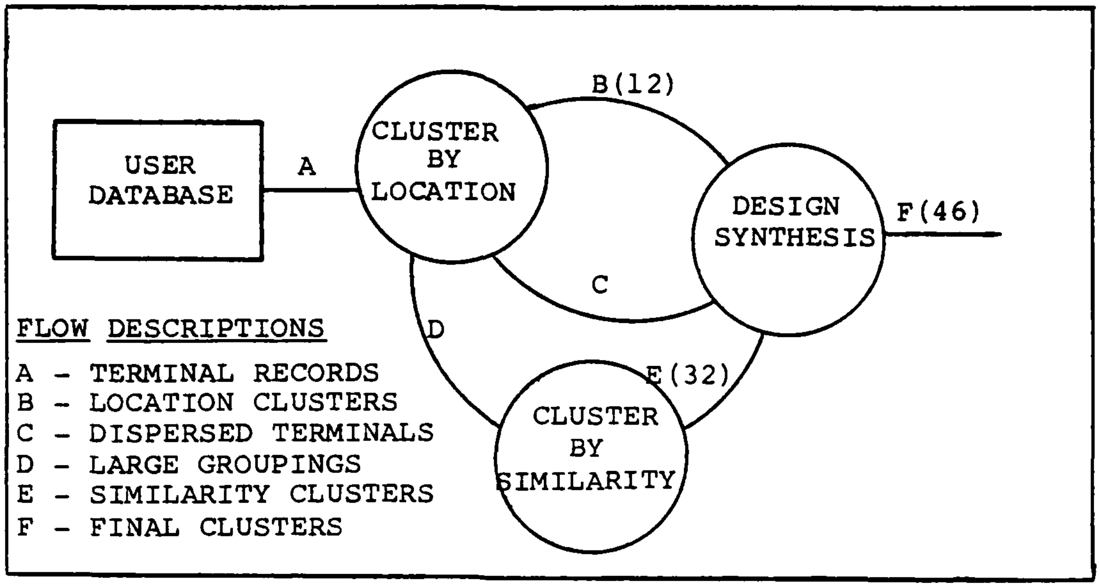

Fig 4-2. Cluster Development DFD

Terminal ID's	Location of User
*B95	Army Personnel Center, Alexandria, Va.
B108	AFTAC/DPM, Patrick AFB, Fla.
B115/G206	Goodfellow AFB, Tx.
*FAO	Air National Guard, White Plains, N.Y.
*G24/G310	AFMEA, Randolph AFB, Tx.
*G29	AFCOMS, Kelly AFB, Tx.
*G50	Naval Observatory, Washington, D.C.
G68/G97/G205	Brooks AFB, Tx.
*G91/G304	Maxwell AFB, Al.
*G302	Air University, Gunter AFS, Al.
G312/G336/G337	Colorado Springs, Col.
G313/G415	Norton AFB, Ca.
*G393	AFIS, Fort Ritchie, Md.
*G394	AFIS, Fort Belvoir, Va.
G395	AFTEC, Kirtland AFB, N.M.
G397	AFISC, Presidio of Monterey, Ca.
G398	AFESC, Tyndall AFB, Fla.
G422/G502	Bergstrom AFB, Tx.
*G500/G504	Dobbins AFB, Ga.
G501	McClellan AFB, Ca.
G503	Air National Guard, Camp Mabry, Tx.

Fig 4-3. Dispersed Terminal Locations

into three separate clusters based on locality was also influenced by the size of each organization's user community and their separate locations on Randolph.

The 12 clusters of acceptable size achieved at this level of the design process are documented as the first 12 clusters in Appendix C. Although cluster cohesion based on similarity values was not a consideration in the design of these clusters, similarity values were computed with average cluster similarities ranging from 100% to 25%. This average cluster similarity value is shown below the similarity tables for each cluster and was computed by obtaining the average similarity of each terminal to all other terminals in the cluster and then averaging these average values. In those instances where terminals in a cluster had no recorded workload for the collection period, the cluster similarity values do not include the impact from such terminals.

The remaining geographical groups of terminals ranged in size from 14 at Wright-Patterson AFB, to 168 terminals at AFMPC. These groups became input to the iterative similarity analysis process. Each iteration of this process consisted of the generation of a similarity table for the cluster so that users would be included in the table only if their similarity value exceeded the target value used as input for the comparison run. The comparisons normally began using a 90% minimum similarity and succeeding runs would be made using lower target levels after removing newly allocated

terminals. In every case, clusters were being established with no more than 12 terminals as required by the criterion threshold established earlier. Eventually, the number of unallocated terminals would be reduced to a number which could be allocated manually, and a single table would be created with no lower limit on the similarity value required. Included in these groups of users were eight which had no recorded workload. Their allocation to a system node was based on the user location or functional affiliation such as the building location or office symbol. Thirty-two additional clusters were built using this iterative process, resulting in a total of 44 clusters created after the application of the locality and similarity value criteria. All of these clusters are also documented in the listing of clusters (Appendix C), with average cluster similarity values computed.

Synthesis

The synthesis portion of the design process serves to perform a "clean-up" function which identifies problem areas in the design and corrects them. The allocation of terminals with no workload could be included as part of this activity, but has been described in the previous section since the locality criterion was met. The major activity accomplished to synthesize the design involved analyzing the remaining unallocated terminals and making a determination as to the proper allocation approach. The 31 terminals involved in

the synthesis activity are listed in Figure 4-3, including their locations. Their wide geographic dispersion suggested that proximity to a nearby cluster would be an appropriate first step, much like the initial allocation criterion employed previously. A total of 13 of these terminals were allocated based on this locality criterion. They are indicated by the asterisk next to their terminal identifiers in the figure. The remaining 18 terminals did not easily conform to this type of allocation because there were two or three terminals in the same area and allocation to nearby nodes would increase the node size beyond the threshold. These terminals were allocated to newly created nodes called CENTRAL1 and CENTRAL2. Their allocation between these two nodes was accomplished by an application of both the locality and similarity value criterion to achieve a "reasonable" balance. The addition of these last two clusters completed the design process with a total of 46 distributed system clusters supporting a total of 425 terminals.

Summary

The distributed system design envisions a total of 46 terminal clusters supporting groups of terminals ranging in size from 3 terminals at the Alaskan Air Command, to 12 terminals in a cluster at Andrews Air Force Base. Although the computed similarity values were important to the partitioning and allocation activities, terminal locations overrode this

factor in many cases. Some of the allocation decisions reached in the synthesis process may, under detailed analysis of communications costs/capabilities, need to be revised. Additionally, expansion of the descriptive data available for use in the generation of similarity values, and functional requirement changes which may occur over time, will necessitate monitoring and possible adjustment to the developed cluster designs.

V Conclusions and Recommendations

Conclusions

The purpose of this investigation was to perform an analysis of the central site portion of the Personnel Data System, propose a distributed processing system design strategy, and develop a design. The analysis performed provided a depiction of current system operations based on the usage of system functions as portrayed in available system documentation reports. The use of system functions was related to the input source system terminals, and these relationships were reflected in a user database described in Chapter IV.

The development of a distributed processing system design strategy evolved from a review of recent publications documenting current approaches in the subject area, and the constraints imposed by current system documentation and the AFMPC hardware reacquisition project. The design strategy described in Chapter IV sought to enable establishment of clusters of system terminals based on a precedence application of three criteria: terminal location, similarity of system function use, and office location/symbol when allocation could not be supported by either of the first two criteria. An ideal cluster size of no more than 12 terminals was established as a criterion threshold which would conform to the equipment constraints imposed by the AFMPC reacquisition program and provide future upgrade potential without

overwhelming equipment capabilities.

Working from the user database containing descriptive information on 425 system terminals, 12 clusters based solely on geographic location emerged. An additional 32 clusters were built based on a similarity analysis within location groupings. Two central clusters at AFMPC were established to support 18 terminals that were not closely situated to any other established cluster locations. The completed distributed system design establishes a total of 46 clusters, each described in Appendix B, including the computed individual and cluster average similarity values.

General Recommendations

One obstacle continually encountered during the analysis phase of the design process was the lack of necessary documentation required to support a more complete design approach. The most complete source of information to depict system function usage was the TERMUSE report which provides usage data for 14 categories of system functions. However, four of these categories include lower levels of usage which are concealed from analysis. In the case of the SURF, ATLAS, and PERSTRANS functions, identification of the specific data file being accessed is essential to an accurate portrayal of the actual operation being performed by the users of each terminal. Likewise, in the case of the "RUN PROG" function category, more than forty different system functions are

actually concealed within this single column of information.

The batch processing nature of the PDS central site removed several levels of complexity from the design strategy, since data concurrency and other problems related to processing in a "real-time" distributed processing system did not have to be accommodated. From the literature review accomplished, the approach employed in response to these problems for SDD-1 was very impressive. The use of conflict graph analysis during the design of a system can significantly reduce software overhead associated with ensuring data integrity and preventing deadlocks across a distributed system. Serious consideration should be given to producing transaction class information in future system development efforts. The availability of such information would be a source of consistant documentation and would serve as a starting point for application of conflict analysis should system evolution proceed towards establishment of a real-time distributed processing system.

Specific Recommendations

Although the intent of this investigation was to ultimately develop a distributed system design which could be implemented at AFMPC, there is considerable additional work that will be required to demonstrate the validity of this or any other distributed design. The following topical areas are of major concern. They require further analysis and

development to enable successful development of a design ready to be implemented.

System Report Requirements. The initial effort does not directly relate to the distributed design. Expansion of system reports to provide further detail of system function usage by terminal users is essential. This documentation would entail expansion of the TERMUSE report to include file usage and further definition of the functions currently included in the "RUN PROG" category of the report. This expansion will provide the basis for generation of more accurate representations of similarities in functional usage for terminals in each cluster.

Data Usage Documentation. The second important effort is the requirement to improve the accuracy of the cluster development process based on inclusion of data usage similarities between the cluster members. This information is only partially available in the current system through the file designation portion of the transaction identifiers shown in the TERMSTRIP report. These file designation were included in the system design developed, but only reflect file usage under the PERSTRANS function. Complete documentation of file usage would support creation of data access matrices described by Palmer (Ref 13:25-26). The system design can then incorporate the proximity measures supported by Palmer.

Transaction Analysis Capabilities. Each transaction within the PDS must be analyzed to accomplish a thorough

conflict analysis process. Identification of transaction classes and placement of the transactions appropriately will be a large undertaking due to the size of the system and the individual sets of transactions for each of the subsystems. This process can be standardized and expedited by development of software that can process the data descriptor table information maintained on disk at AFMPC. These tables could be obtained in either magnetic tape or card form very readily, and could be used as input to routines which would identify transaction attributes and assign each transaction to an appropriate class.

Network Analysis. Communications within the distributed processing system is a vital concern, yet has been largely ignored in this thesis because of a lack of familiarity with the subject and the time required to perform a reasonable network analysis. The only consideration of communications emerged in the selection of user location as a primary criterion for cluster allocations. Therefore, a major effort yet to be accomplished is the analysis of the distributed design on the basis of network capabilities and costs. As mentioned earlier in Chapter II, such considerations should be included in the analysis, partitioning, allocation, and synthesis processes employed to achieve the distributed design.

System Simulation. The final recommendation for future

work that would contribute to improvement and validation of the design strategy concerns the development of a simulation package. Prior to starting this thesis, the author undertook two simulation projects which sought to simulate central site operations both in the current environment, and as the system might exist under a distributed processing system structure. These efforts encountered many of the same problems with shortcomings in detailed documentation required to perform the level of analysis necessary for development of the design strategy. Information to show current system performance characteristics are not readily available and would require the installation of software traps to collect data in most cases. Such traps could be developed and used periodically with the assistance of system managers at AFMPC, but the benefits to be derived must justify such changes.

Any simulation package developed will be subject to the problems of size since the transaction processing portion of the system involves many files, several hundred terminals, and numerous unique application programs which continually compete for system resources. Some of the specific information requirements identified as necessary for development of a simulation package include the following:

- Processing times (CPU) required for the various segments of transaction processing activities must be obtained for the various application programs.

- Transaction input rates, reject rates, and re-input rates for system users must be collected and measured for

each of various system functions based on such input (i.e. SURF, PERSTRANS, and ATLAS).

- Development of descriptions of the user/system interaction for the various system functions.

- Resource requirements and utilization statistics for the various system functions.

Summary

The distributed system design developed is an initial effort towards development of a design which could be implemented. The size of the current system, the need for expansion of descriptive system reports, and collection of detailed system statistics to support simulation of system performance to validate the design, provides the opportunity for continuation of this project in a rapidly emerging technological area. Early contact with AFMPC/MPCD, the sponsor, must be established to develop a strategy for implementing the system information sources required. Additionally, any individual interested in undertaking the project should have experience with an available simulation language such as Q-GERT or SLAM.

Bibliography

1. AFMPC Competitive Reacquisition Transition Plan. Randolph AFB, Texas: Air Force Manpower and Personnel Center, 8 July 1980.

2. Bernstein, Philip A., David W. Shipman, and James B. Rothnie, Jr. "Concurrency Control in a System for Distributed Databases (SDD-1)," ACM Transactions on Database Systems, 5(1): 18-51 (March 1980).

3. Bernstein, Philip A., James B. Rothnie, Jr., Nathan Goodman, and Christos A. Papadimitriou. "The Concurrency Control Mechanism of SDD-1: A System for Distributed Databases (The Fully Redundant Case)," IEEE Transactions on Software Engineering, 4(3): 154-168 (May 1978).

4. Booth, Grayce M. The Distributed System Environment. New York: McGraw-Hill Book Company, 1981.

5. Broadstreet, Maj Ralph. AFMPC Competitive Reacquisition Action Officer, Air Force Manpower and Personnel Center (personal interview). Randolph AFB, Texas, September 1981.

6. Buckles, B. P. and D. M. Hardin. "Partitioning and Allocation of Logical Resources in a Distributed Computing Environment," Tutorial: Distributed System Design, Chapter 3. IEEE Computer Society, 1979.

7. Chu, Wesley W. "Optimal File Allocation in a Multiple Computer System," Tutorial: Centralized and Distributed Data Base Systems, 414-415. IEEE Computer Society, 1979.

8. Date, C. J. An Introduction to Database Systems. Philippines: Addison-Wesley Publishing Company, 1977.

9. Donaldson, Hamish. Designing a Distributed Processing System. New York: John Wiley and Sons, 1979.

10. Lelann, Gerard. "A Distributed System for Real-Time Transaction Processing," Computer, 4: 43-48 (February 1981).

11. ----- "An Analysis of Different Approaches to Distributed Computing," Tutorial: Centralized and Distributed Data Base Systems, 222,231. IEEE Computer Society, 1979.

12. Martin, James. Design and Strategy for Distributed Processing. Englewood Cliffs, N.J.: Prentice Hall, 1981.

13. Palmer, D. F. "Distributed Computing System Design at the Subsystem/Network Level," Tutorial: Centralized and Distributed Data Base Systems, 22-30. IEEE Computer Society, 1979.

14. Patrick, Robert I. Application Design Handbook for Distributed Systems. Boston: CBI Publishing Company, 1980.

15. Peebles, Richard and Eric Manning. "System Architecture for Distributed Data Management," Tutorial: Centralized and Distributed Data Base Systems, 351-357. IEEE Computer Society, 1979.

16. Rothnie, J. B. Jr., P. A. Bernstein, S. Fox, N. Goodman, M. Hammer, T. A. Landers, C. Reeve, D. W. Shipman, and E. Wong. "Introduction to a System For Distributed Databases (SDD-1)," ACM Transactions on Database Systems, 5(1): 1-17 (March 1980).

17. Thomas, Robert H. "A Solution to the Concurrency Control Problem for Multiple Copy Databases," Tutorial: Centralized and Distributed Data Base Systems, 509-515. IEEE Computer Society, 1979.

18. Ullman, Jeffrey D. Principles of Database Systems. Rockville, Md.: Computer Science Press, 1980.

19. Weinberg, Victor. Structured Analysis. New York: Yourdon Press, 1978.

20. Yourdon, Edward and Larry L. Constantine. Structured Design. Englewood Cliffs, N.J.: Prentice-Hall, 1979.

Appendix A

Central Site Environment

The central site consists of two B6700 computers (A system and B system) and one H6068 computer (C system).

1. The A system consists of the following components:

   a. 3 central processors
   b. 960K memory
   c. 4 data communications processors
   d. 14 tape drives
   e. 136 disk drives (dedicated, switchable, and shared)

2. The B system consists of the following components:

   a. 3 central processors
   b. 786K memory
   c. 2 data communication processors
   d. 15 tape drives
   e. 162 disk drives (dedicated, switchable, and shared)

   NOTE: Total disk drives = 236

3. The C system consists of the following components:

   a. 1 central processor
   b. 256K memory
   c. 1 data communications processor
   d. 5 tape drives
   e. 14 disk drives

A total of 502 permanent terminals were identified from system documentation gathered for use in this thesis. Of this number, 77 terminals were employed by PROMIS/PMS users who were excluded from the distributed system design. An additional 55 terminals, not part of the 502, are portables which could not be reliably reflected as belonging to any specific user consistently.

APPENDIX B

Database Software Listing

```
(*$R- *)
PROGRAM MANAGER(INPUT/,OUTPUT,DATAFIL,MTRIXFIL) ;
(**************************************************************)
(*
(*   PROGRAM MANAGER WAS DEVELOPED BY CAPT. RONALD V. BRANDT
(*   AT THE AIR FORCE INSTITUTE OF TECHNOLOGY AS SUPPORTING
(*   SOFTWARE FOR THE DEVELOPMENT OF A DISTRIBUTED PROCESSING
(*   SYSTEM DESIGN THESIS PROJECT.
(*
(*   FUNCTION: THIS PROGRAM SUPPORTS THE CREATION, MAINTENANCE,
(*   AND ANALYSIS OF A DATABASE.  AS PRESENTED, THIS DATABASE
(*   CONSISTS OF USER/TERMINAL RECORDS WHICH ARE DEFINED BY
(*   THE SYSTEM TRANSACTION/FUNCTION RECORDS LINKED TO IT.
(*   THE MENU OF AVAILABLE PROGRAM COMMANDS ARE LISTED BELOW:
(*
(*                READ INPUT     SUBCLUSTER
(*                SAVE DATA      CLUSTER
(*                ADD USER       COMPARE
(*                DEL USER       SORT USERS
(*                ADD TRANS      ADD OFFICE
(*                DEL TRANS      END
(*
(*   DOCUMENTATION:  EACH OF THE COMMANDS ARE DOCUMENTED AT
(*   THE BEGINNING OF THE APPROPRIATE SECTION OF CODE.
(*
(*   INPUT:  THIS PROGRAM OPERATES IN AN INTERACTIVE MODE
(*   BY PRESENTING THE USER WITH THE MENU LIST AND PROMPTING
(*   ANY INPUT REQUIRED.  THE USE OF THIS PROGRAM REQUIRES
(*   THE PRESENCE OF A LOCAL FILE CALLED 'DATAFIL' OR INITIAL
(*   CREATION OF THE FILE THROUGH USE OF THE 'ADD USER', 'ADD
(*   TRANS', AND THE 'SAVE DATA' COMMANDS.
(*
(*   OUTPUT:  THREE TYPES OF OUTPUTS OCCUR AS A RESULT OF THE
(*   USE OF THIS PROGRAM.  INTERACTIVE MESSAGES ARE DISPLAYED
(*   ON THE USERS TERMINAL;  OUTPUT OF THE FILE 'DATAFIL'
(*   IS ACCOMPLISHED UPON INPUT OF THE 'SAVE DATA' COMMAND;
(*   AND OUTPUT OF THE FILE 'MTRIXFI' OCCURS AS A RESULT OF
(*   INPUT OF THE 'COMPARE' COMMAND.
(*
(**************************************************************)
```

```
LABEL
   10 , 20, 80, 85,
   75, 99 ;


TYPE
   TRANSRECORD = ^TRANSENTRY ;
   TRANSENTRY = RECORD
      TRANSNAME : ALFA ;
      NEXTTRANS : TRANSRECORD ;
    END ;
   USERRECORD = ^USERENTRY ;
   USERENTRY = RECORD
      NAME : ALFA ;
      NEXT : USERRECORD ;
      OFFICE : ALFA ;
      USERTRANS : TRANSRECORD ;
      TRANSCOUNT : INTEGER ;
    END  ;


VAR
   MTRIXFIL ,
   DATAFIL : TEXT ;
   TRANSPOINTER : TRANSRECORD ;
   USERPOINTER,
   USERHEAD : USERRECORD ;
   CHANGE : BOOLEAN ;
   COMMAND,
   OFFICEID ,
   USERID,
   TRANSID : ALFA ;
   CLUSTER : ARRAY[1..200] OF USERRECORD ;
   USERCOUNT : INTEGER ;
```

```
(***********************************************************)
(*
(* PROCEDURE GETFROMFILE IS USED TO READ THE NEXT ALFA ENTRY
(* FROM THE INPUT FILE "DATAFIL."  EACH ALFA ENTRY CONSISTS
(* OF TEN CHARACTERS, AND THE TEN CHARACTERS READ ARE RE-
(* TURNED TO THE CALLER THROUGH THE VARIABLE INCHAR.  CALLING
(* FORMAT : GETFROMFILE(X); WHERE X HAS BEEN DECLARED AS A
(* VARIABLE OF DATA TYPE ALPHA.
(*
(***********************************************************)

PROCEDURE GETFROMFILE(VAR INCHAR : ALFA) ;
VAR
   I : INTEGER ;
   EMPTY : ALFA ;
BEGIN
   EMPTY := '          ' ;
   IF EOLN(DATAFIL) THEN BEGIN
      READLN(DATAFIL) ;
      INCHAR := EMPTY ;
   END
   ELSE BEGIN
     FOR I := 1 TO 10 DO
        IF EOLN(DATAFIL) THEN INCHAR[I] := ' '
        ELSE READ(DATAFIL,INCHAR[I]) ;
   END ;
END ;

(************************************************************)
(*
(* PROCEDURE GETALFA IS USED TO READ AN ALFA ENTRY TYPED IN
(* OVER THE KEYBOARD CALLING FORMAT : GETALFA(X); WHERE X
(* HAS BEEN DECLARED AS A VARIABLE OF DATA TYPE ALFA.
(*
(************************************************************)

PROCEDURE GETALFA(VAR INALFA : ALFA) ;
VAR
   I : INTEGER ;
BEGIN
   FOR I := 1 TO 10 DO
      IF EOLN(INPUT) THEN INALFA[I] := ' '
      ELSE READ (INALFA[I]) ;
END  ;
```

```
(*************************************************************)
(*
(* PROCEDURE GETCOMMAND DISPLAYS ON THE TERMINAL THE MENU
(* OF AVAILABLE PROGRAM COMMANDS AND CALLS PROCEDURE GETALFA
(* TO GET THE USERS SELECTED COMMAND.
(* CALLING FORMAT : GETCOMMAND(X); WHERE X HAS BEEN DECLARED
(* AS A VARIABLE DATA TYPE ALFA.
(*
(*************************************************************)

PROCEDURE GETCOMMAND(VAR NEWCOMMAND : ALFA) ;
BEGIN
   WRITELN(' ENTER COMMAND - ADD USER') ;
   WRITELN('               - ADD TRANS') ;
   WRITELN('               - READ INPUT') ;
   WRITELN('               - SAVE DATA') ;
   WRITELN('               - ADD OFFICE') ;
   WRITELN('               - DEL TRANS') ;
   WRITELN('               - DEL USER') ;
   WRITELN('               - END') ;
   WRITELN('               - CLUSTER') ;
   WRITELN('               - SUBCLUSTER') ;
   WRITELN('               - SORT USERS') ;
   WRITELN('               - COMPARE') ;
   WRITELN ;
   READLN ;
   GETALFA(NEWCOMMAND) ;
END ;

(*************************************************************)
(*
(* PROCEDURE PUTALFA WRITES TO THE TEXTFILE DATAFIL THE ALFA
(* DATA ITEM PASSED TO IT IN THE CALL.
(* CALLING FORMAT : PUTALFA(X) ; WHERE X HAS BEEN DECLARED
(* AS A VARIABLE OF DATA TYPE ALFA.
(*
(*************************************************************)

PROCEDURE PUTALFA(OUTALFA : ALFA) ;
VAR
   I : INTEGER ;
BEGIN
   WRITE(DATAFIL, OUTALFA) ;
END ;
```

```
(**********************************************************)
(*
(* PROCEDURE SAVEDATA WRITES THE CONTENTS OF THE USER DATA-
(* BASE TO THE OUTPUT FILE DATAFIL.  OUTPUT BEGINS WITH THE
(* USER POINTED TO BY USERHEAD AND CONTINUES UNTIL ALL USERS
(* AND THEIR DATA HAVE BEEN WRITTEN.  TO ALLOW DISPLAY OF THE
(* OUTPUT FILE ON A TERMINAL SCREEN, EACH LINE OF THE FILE
(* HAS NO MORE THAN 80 CHARACTERS OF DATA.  THE NAME,
(* OFFICE ID (IF FIRST LINE FOR THAT USER), THE COUNT OF THE
(* NUMBER OF SYSTEM FUNCTIONS/TRANSACTIONS ON THE REMAINDER
(* OF THE LINE, AND THE FUNCTION OR TRANSACTION ID'S.  A
(* COUNT OF THE NUMBER OF USERS WRITTEN TO THE FILE IS OUT-
(* PUT TO THE TERMINAL AT THE COMPLETION OF THE SAVE PRO-
(* CEDURE.  CALLING FORMAT : SAVEDATA ;
(*
(**********************************************************)

PROCEDURE SAVEDATA ;
CONST
   MAXCOUNT = 5 ;
VAR
   CH : CHAR ;
   TEMPCOUNT, J : INTEGER ;
   TEMPUSER, CURRENTUSER : USERRECORD ;
   CURRENTTRANS : TRANSRECORD ;
BEGIN
   REWRITE(DATAFIL) ;
   CH := ' ' ;
   CURRENTUSER := USERHEAD ;
   WRITELN(' INCOMING USER COUNT: ',USERCOUNT : 3) ;
   USERCOUNT := 0 ;
   WHILE CURRENTUSER <> NIL DO
      BEGIN
         TEMPUSER := CURRENTUSER ;
         USERCOUNT := USERCOUNT + 1 ;
         TEMPCOUNT := CURRENTUSER^.TRANSCOUNT ;
         CURRENTTRANS := TEMPUSER^.USERTRANS ;
         WRITE(DATAFIL,CURRENTUSER^.NAME) ;
         WRITE(DATAFIL,CURRENTUSER^.OFFICE) ;
         IF TEMPCOUNT = 0 THEN WRITE(DATAFIL,' 0 ') ;
         WHILE TEMPCOUNT > 0 DO BEGIN
           IF TEMPCOUNT > MAXCOUNT THEN
              BEGIN
                 TEMPCOUNT := TEMPCOUNT - MAXCOUNT ;
                 WRITE(DATAFIL, MAXCOUNT:1) ;
                 FOR J := 1 TO MAXCOUNT DO BEGIN
                    WRITE(DATAFIL,CURRENTTRANS^.TRANSNAME) ;
                    CURRENTTRANS := CURRENTTRANS^.NEXTTRANS ;
                    END ;
                 WRITELN(DATAFIL) ;
                 WRITE(DATAFIL,CURRENTUSER^.NAME) ;
              END
```

```
                  ELSE BEGIN
                     WRITE(DATAFIL,' ',TEMPCOUNT:1,' ') ;
                     FOR J := 1 TO TEMPCOUNT DO BEGIN
                        WRITE(DATAFIL,CURRENTTRANS^.TRANSNAME) ;
                        CURRENTTRANS := CURRENTTRANS^.NEXTTRANS ;
                        END ;
                     TEMPCOUNT := 0 ;
                  END ; (* ELSE *)
                END ; (* WHILE *)
             CURRENTUSER := TEMPUSER^.NEXT ;
             WRITELN(DATAFIL) ;
          END ;
       WRITELN(DATAFIL,'END         ') ;
       RESET(DATAFIL) ;
       WRITELN(' OUTPUT USER COUNT: ',USERCOUNT : 3) ;
    END ;
```

```
(*****************************************************************)
(*
(* PROCEDURE GETOFFICEID IS USED TO REQUEST INPUT OF A USERS
(* OFFICEID FROM THE TERMINAL.
(* CALLING FORMAT: GETOFFICEID(X) ;
(* WHERE X HAS BEEN DECLARED AS A VARIABLE OF DATA TYPE ALFA.
(*
(*****************************************************************)

PROCEDURE GETOFFICEID(VAR NEWOFFICE:ALFA) ;
BEGIN
   WRITELN(' ENTER OFFICE ID OR END ') ;
   READLN ;
   GETALFA(NEWOFFICE) ;
END ;

(*****************************************************************)
(*
(* PROCEDURE GETUSERID IS USED TO REQUEST INPUT OF A USERID
(* FROM THE TERMINAL.  CALLING FORMAT : GETUSERID(X) ; WHERE
(* X HAS BEEN DECLARED AS A VARIABLE OF DATA TYPE ALFA.
(*
(*****************************************************************)

PROCEDURE GETUSERID(VAR NEWUSER : ALFA) ;
BEGIN
   WRITELN(' ENTER USER ID OR END');
   READLN ;
   GETALFA(NEWUSER) ;
END ;
```

```
(**********************************************************)
(*
(* FUNCTION DUPLICATEUSER IS A BOOLEAN FUNCTION USED TO DETER-
(* MINE IF THE USERID PASSED TO THE FUNCTION IS CURRENTLY
(* PART OF THE USER DATABASE.  IF A MATCH OF THE CHECKID AND
(* CURRENT USERID IS FOUND, DUPLICATEUSER IS SET TRUE AND
(* THE RECORD POINTER IS RETURNED POINTING TO THE MATCHED
(* RECORD.  CALLING FORMAT : IF DUPLICATEUSER (X,Y) THEN ;
(* WHERE X IS A VARIABLE OF DATA TYPE ALFA CONTAINING THE
(* USERID TO CHECKID, AND Y HAS BEEN DECLARED AS A VARIABLE
(* OF DATA TYPE USERRECORD.
(*
(**********************************************************)

FUNCTION DUPLICATEUSER(CHECKID:ALFA;VAR USERNAME:USERRECORD)
                              :BOOLEAN ;
VAR
   TEMPCHECK : BOOLEAN ;
BEGIN
  TEMPCHECK := FALSE ;
  IF USERHEAD <> NIL THEN BEGIN
   USERNAME := USERHEAD ;
   TEMPCHECK := USERNAME^.NAME = CHECKID ;
   WHILE (USERNAME^.NAME <> CHECKID) AND
         (USERNAME^.NEXT <> NIL) DO
      BEGIN
         USERNAME := USERNAME^.NEXT ;
         TEMPCHECK := USERNAME^.NAME = CHECKID ;
      END ;
  END ;
 DUPLICATEUSER := TEMPCHECK ;
END ;
```

```
(**********************************************************)
(*
(* PROCEDURE ADDOFFICE INSERTS THE OFFICEID (PASSED TO IT)
(* INTO ALL USER RECORDS SPECIFIED FROM THE TERMINAL.
(* TERMINAL INPUT OF USERID IS REQUESTED AND THE OFFICEID IS
(* INSERTED UNTIL THE COMMAND "END" IS INPUT.
(* CALLING FORMAT : ADDOFFICE(X) ; WHERE X IS THE OFFICEID
(* VARIABLE OF DATA TYPE ALFA.
(*
(**********************************************************)

PROCEDURE ADDOFFICE(USEROFFICE:ALFA) ;
BEGIN
   WRITELN(' ENTER USERID FOR OFFICE, OR END ') ;
   READLN ;
   GETALFA(USERID) ;
   WHILE NOT (USERID = 'END       ') DO BEGIN
      IF (DUPLICATEUSER(USERID,USERPOINTER)) THEN
         USERPOINTER^.OFFICE := USEROFFICE ;
      WRITELN(' ENTER USERID FOR OFFICE, OR END ') ;
      READLN ;
      GETALFA(USERID) ;
   END ;
END ;

(**********************************************************)
(*
(* PROCEDURE DELETEUSER IS USED TO REMOVE A USER RECORD
(* FROM THE USER DATABASE.
(* CALLING FORMAT : DELETEUSER(X) ; WHERE X IS A VARIABLE OF
(* DATA TYPE ALFA CONTAINING THE USERID TO BE DELETED.
(*
(**********************************************************)

PROCEDURE DELETEUSER(OLDUSER : USERRECORD) ;
VAR
   TEMPUSER : USERRECORD ;
BEGIN
   TEMPUSER := USERHEAD ;
   IF TEMPUSER = OLDUSER THEN BEGIN
      USERHEAD := TEMPUSER^.NEXT ;
      DISPOSE(OLDUSER) ;
   END
   ELSE BEGIN
      WHILE (TEMPUSER^.NEXT<>OLDUSER) AND
            (TEMPUSER^.NEXT <> NIL) DO
         TEMPUSER := TEMPUSER^.NEXT ;
      IF TEMPUSER^.NEXT = OLDUSER THEN BEGIN
         TEMPUSER^.NEXT := OLDUSER^.NEXT ;
         DISPOSE(OLDUSER) ;
      END ;  (* IF *)
   END ;
END ;
```

```
(**************************************************************)
(*
(* PROCEDURE BUILDUSER CREATES USER RECORDS FOR THE USERIDS
(* BROUGHT TO THE PROCEDURE, AND LINKS THE RECORD INTO THE
(* DATABASE.  THE VARIABLES USERPOINTER AND USERHEAD POINT
(* TO THE NEW RECORD AT THE COMPLETION OF THE PROCEDURE.
(* CALLING FORMAT : BUILDUSER(X) ; WHERE X IS A VARIABLE OF
(* DATA TYPE ALFA CONTAINING THE NEW USERID.
(*
(**************************************************************)

PROCEDURE BUILDUSER(NEWID : ALFA) ;
BEGIN
   NEW(USERPOINTER) ;
   USERPOINTER^.NEXT := USERHEAD ;
   USERHEAD := USERPOINTER ;
   USERPOINTER^.NAME := NEWID ;
   USERHEAD^.USERTRANS := NIL ;
   USERHEAD^.TRANSCOUNT := 0 ;
   USERHEAD^.OFFICE := '          ' ;
   USERHEAD^.CLUSTERMEMBER := FALSE ;
END ;

(**************************************************************)
(*
(* PROCEDURE BUILDTRANS CREATES NEW TRANSACTION RECORDS,
(* AND LINKS THEM TO THE USER RECORD IN PARENTPOINTER.  THE
(* TRANSACTION RECORD IS POINTED TO BY THE GLOBAL VARIABLE
(* TRANSPOINTER UPON COMPLETION OF THE PROCEDURE.
(* CALLING FORMAT : BUILDTRANS(X,Y) ; WHERE X IS AN ALFA
(* VARIABLE CONTAINING THE TRANSACTION ID, AND Y IS OF
(* TYPE USERRECORD POINTING TO THE USER FOR THIS TRANS-
(* ACTION.
(*
(**************************************************************)

PROCEDURE BUILDTRANS(ADDTRANS:ALFA;PARENTPOINTER:USERRECORD);
BEGIN
   NEW(TRANSPOINTER) ;
   TRANSPOINTER^.TRANSNAME := ADDTRANS ;
   TRANSPOINTER^.NEXTTRANS := PARENTPOINTER^.USERTRANS ;
   PARENTPOINTER^.USERTRANS := TRANSPOINTER ;
   PARENTPOINTER^.TRANSCOUNT :=PARENTPOINTER^.TRANSCOUNT+1;
END ;
```

```
(***********************************************************)
(*
(* PROCEDURE SORTUSERS IS USED TO SORT THE USER DATABASE IN
(* ASCENDING ORDER BASED ON USERID (NAME).  THE ALGORITHM
(* IMPLEMENTED IS AN EXCHANGE SELECTION SORT. THIS PROCEDURE
(* IS INVOKED BY INPUT OF THE "SORT USERS" COMMAND.
(* CALLING FORMAT : SORTUSERS ;
(*
(***********************************************************)

PROCEDURE SORTUSERS ;
VAR
   PREVPOINTER ,
   TOPPOINTER, CURRPOINTER : USERRECORD ;
   CHANGE : BOOLEAN ;

PROCEDURE FINDSMALLEST ;
BEGIN
   PREVPOINTER := USERPOINTER ;
   WHILE CURRPOINTER <> NIL DO BEGIN
      IF (CURRPOINTER^.NAME < USERPOINTER^.NAME) THEN BEGIN
         TOPPOINTER^.NEXT := CURRPOINTER ;
         PREVPOINTER^.NEXT := CURRPOINTER^.NEXT ;
         CURRPOINTER^.NEXT := USERPOINTER ;
         CURRPOINTER := USERPOINTER ;
         USERPOINTER := TOPPOINTER^.NEXT ;
         CHANGE := TRUE ;
      END
      ELSE PREVPOINTER := CURRPOINTER ;
   CURRPOINTER := CURRPOINTER^.NEXT ;
   END ; (* WHILE *)
END ; (* FINDSMALLEST *)

BEGIN
   TOPPOINTER := USERHEAD ;
   CURRPOINTER := USERHEAD^.NEXT ;
   IF (CURRPOINTER^.NAME < USERHEAD^.NAME) THEN BEGIN
      USERHEAD := CURRPOINTER ;
      TOPPOINTER^.NEXT := CURRPOINTER^.NEXT ;
      CURRPOINTER^.NEXT := TOPPOINTER ;
    END ;
   TOPPOINTER := USERHEAD ;
   USERPOINTER := USERHEAD^.NEXT ;
   CURRPOINTER := USERPOINTER^.NEXT ;
   FINDSMALLEST ;
   IF (TOPPOINTER^.NAME > USERPOINTER^.NAME) THEN BEGIN
      USERHEAD := USERPOINTER ;
      TOPPOINTER^.NEXT := USERPOINTER^.NEXT ;
      USERPOINTER^.NEXT := TOPPOINTER ;
    END ;
   TOPPOINTER := USERHEAD ;
   USERPOINTER := USERHEAD^.NEXT ;
```

```
    CURRPOINTER := USERPOINTER^.NEXT ;
    CHANGE := TRUE ;
    WHILE (CURRPOINTER^.NEXT<>NIL) DO BEGIN
      CHANGE := FALSE ;
      FINDSMALLEST ;
      TOPPOINTER := TOPPOINTER^.NEXT ;
      USERPOINTER := TOPPOINTER^.NEXT ;
      CURRPOINTER := USERPOINTER^.NEXT ;
     END ;
END ;
```

```
(**************************************************************)
(*
(* PROCEDURE READOLDFILE IS USED TO READ THE EXISTING USER
(* DATABASE FROM FILE DATAFIL, AND CREATE THE LINKED DATA
(* STRUCTURE EMPLOYED FOR FILE MAINTENANCE. THE NUMBER
(* OF USER RECORDS READ IN IS DISPLAYED ON THE TERMINAL UPON
(* COMPLETION.  CALLING FORMAT : READOLDFILE ;
(* PROCEDURE IS USED IN RESPONSE TO THE "READ INPUT" COMMAND.
(*
(**************************************************************)

PROCEDURE READOLDFILE ;
LABEL 40,50 ;
VAR CURRENTUSER : ALFA ;
   J : INTEGER ;
   CH : CHAR ;
   TEMPCOUNT : INTEGER ;
BEGIN
   RESET(DATAFIL) ;
   USERCOUNT := 0 ;
   CH := ' ' ;
   CURRENTUSER := '          ' ;
   WHILE NOT EOF(DATAFIL) DO
      BEGIN
         GETFROMFILE(USERID) ;
         IF USERID = CURRENTUSER THEN GOTO 40 ;
         CURRENTUSER := USERID ;
         IF USERID = 'END       ' THEN BEGIN
            WRITELN(' REACHED END OF INPUT FILE.') ;
            GOTO 50 ; END
         ELSE BUILDUSER(USERID) ;
         USERCOUNT := USERCOUNT + 1 ;
         GETFROMFILE(USERPOINTER^.OFFICE) ;
40:      READ(DATAFIL,CH,TEMPCOUNT) ;
         IF TEMPCOUNT <> 0 THEN READ(DATAFIL,CH) ;
         WHILE TEMPCOUNT <> 0 DO BEGIN
            TEMPCOUNT := TEMPCOUNT - 1 ;
            GETFROMFILE(TRANSID) ;
            BUILDTRANS(TRANSID,USERPOINTER) ;
            END ;
         READLN(DATAFIL) ;
      END ;
50:  WRITELN(' COUNT OF INCOMING USERS: ',USERCOUNT : 3) ;
    END ;
```

```
(***************************************************************)
(*
(* PROCEDURE GETTRANSID REQUESTS INPUT OF A TRANSACTION
(* IDENTIFIER AND INVOKES PROCEDURE GETALFA TO READ KEYBOARD
(* INPUT INTO THE ALFA VARIABLE NEWTRANS.
(* CALLING FORMAT: GETTRANSID(X) :
(* WHERE X IS A VARIABLE OF DATA TYPE ALFA.
(*
(***************************************************************)

PROCEDURE GETTRANSID(VAR NEWTRANS : ALFA) ;
BEGIN
   WRITELN(' ENTER TRANSACTION ID OR "END"') ;
   READLN ;
   GETALFA(NEWTRANS) ;
END ;

(***************************************************************)
(*
(* BOOLEAN FUNCTION DUPLICATETRANS IS USED TO DETERMINE
(* WHETHER THE USER POINTED TO BY THE VARIABLE BROUGHT TO
(* THE FUNCTION CURRENTLY HAS THE TRANSACTION IDENTIFIER ALSO
(* BROUGHT TO THE FUNCTION.
(* CALLING FORMAT : IF DUPLICATETRANS(X,Y) THEN ; WHERE X IS
(* AN ALFA VARIABLE CONTAINING THE TRANSACTION IDENTIFIER
(* TO BE CHECKED FOR, AND Y POINTING TO THE USER RECORD TO
(* BE SEARCHED.
(*
(***************************************************************)

FUNCTION DUPLICATETRANS(CHECKID:ALFA;CURRUSER:USERRECORD)
                              : BOOLEAN ;
VAR
   TEMPTRAN : TRANSRECORD ;
   I : INTEGER ;
   FOUND : BOOLEAN ;
BEGIN
   TEMPTRAN := CURRUSER^.USERTRANS ;
   FOUND := FALSE ;
   WHILE NOT (FOUND) AND (TEMPTRAN<>NIL) DO BEGIN
      FOUND := TEMPTRAN^.TRANSNAME = CHECKID ;
      TEMPTRAN := TEMPTRAN^.NEXTTRANS ;
   END ;
   DUPLICATETRANS := FOUND ;
END ;
```

```
(*************************************************************)
(*
(* PROCEDURE BUILDMATRIX READS THE USER DATABASE COMPARING
(* THE TRANSACTIONS AND SYSTEM FUNCTIONS USERS HAVE IN
(* COMMON TO COMPUTE A SIMILARITY VALUE.  THIS VALUE
(* AND THE USER NAMES ARE WRITTEN TO FILE MTRIXFIL IF THE
(* SIMILARITY VALUE IS GREATER THAN THE CUTOFF VALUE(SIMLIMIT).
(* THIS PROCEDURE IS INVOKED IN INPUT OF THE "COMPARE" COMMAND.
(*
(*************************************************************)

PROCEDURE BUILDMATRIX ;
CONST
   SIMLIMIT = 0.5 ;
LABEL
   11, 12, 13 ;
VAR
   MATCHPOINTER : USERRECORD ;
   TOTALCOUNT ,
   COMMONCOUNT, J : INTEGER ;
   SIMILARITY : REAL ;
BEGIN
   REWRITE(MTRIXFIL) ;
   WRITELN(MTRIXFIL,'              USER SIMILARITY TABLE -',
          USERHEAD^.OFFICE) ;
   USERPOINTER := USERHEAD ;
   WHILE USERPOINTER <> NIL DO BEGIN
      IF (USERPOINTER^.CLUSTERMEMBER) THEN GOTO 12 ;
      MATCHPOINTER := USERHEAD ;
      WRITE(MTRIXFIL,USERPOINTER^.NAME) ;
      IF USERPOINTER^.TRANSCOUNT = 0 THEN BEGIN
         WRITE(MTRIXFIL,' NO TRANSACTIONS.') ;
         GOTO 11 ;
         END ;
      WRITELN(MTRIXFIL) ;
      J := 0 ;
      WHILE MATCHPOINTER <> NIL DO BEGIN
         IF (MATCHPOINTER^.CLUSTERMEMBER) THEN GOTO 13 ;
         IF MATCHPOINTER = USERPOINTER THEN
            COMMONCOUNT := USERPOINTER^.TRANSCOUNT
         ELSE BEGIN
            TRANSPOINTER := USERPOINTER^.USERTRANS ;
            COMMONCOUNT := 0 ;
            WHILE TRANSPOINTER <> NIL DO BEGIN
               IF (DUPLICATETRANS(TRANSPOINTER^.TRANSNAME,
                  MATCHPOINTER))
                  THEN COMMONCOUNT := COMMONCOUNT + 1 ;
               TRANSPOINTER := TRANSPOINTER^.NEXTTRANS ;
            END ; (* WHILE *)
         END ; (* ELSE *)
        TOTALCOUNT := USERPOINTER^.TRANSCOUNT +
                      MATCHPOINTER^.TRANSCOUNT ;
        SIMILARITY := COMMONCOUNT/(TOTALCOUNT-COMMONCOUNT) ;
```

```
          IF SIMILARITY > SIMLIMIT THEN BEGIN
             WRITE(MTRIXFIL,MATCHPOINTER^.NAME,
                   SIMILARITY:4:2,'      ') ;
            J := J + 1 ;
          END ;
            IF J = 4 THEN BEGIN
               WRITELN(MTRIXFIL) ;
               J := 0 ;
               END ;
13:         MATCHPOINTER := MATCHPOINTER^.NEXT ;
         END ; (* WHILE *)
11:      WRITELN(MTRIXFIL) ;
12:      USERPOINTER := USERPOINTER^.NEXT ;
      END ; (* WHILE *)
END ;
```

```
(**************************************************************)
(*
(* PROCEDURE DELETETRANS DISPOSES OF TRANSACTION RECORDS
(* BELONGING TO A SPECIFIC USER RECORD.
(* CALLING FORMAT : DELETETRANS(X,Y) ; WHERE X IS A VARIABLE
(* CONTAINING THE USER RECORD POINTER FROM WHICH THE TRANS-
(* ACTION IS TO BE DELETED AND Y IS THE TRANSACTION ID.
(*
(**************************************************************)

PROCEDURE DELETETRANS(OLDUSER: USERRECORD;CHECKID:ALFA) ;
   VAR
      FOUND : BOOLEAN ;
      PREVIOUSTRANS, TEMPTRANS: TRANSRECORD ;
   BEGIN
      TEMPTRANS := OLDUSER^.USERTRANS ;
      IF TEMPTRANS <> NIL THEN BEGIN
         IF TEMPTRANS^.TRANSNAME = CHECKID THEN BEGIN
            OLDUSER^.USERTRANS := TEMPTRANS^.NEXTTRANS ;
            WRITELN(' TRANSACTION DELETED- ',CHECKID) ;
            OLDUSER^.TRANSCOUNT := OLDUSER^.TRANSCOUNT - 1 ;
            DISPOSE(TEMPTRANS) ; END
         ELSE BEGIN
            FOUND := FALSE ;
            WHILE (TEMPTRANS<>NIL) AND (FOUND<>TRUE) DO BEGIN
              PREVIOUSTRANS := TEMPTRANS ;
              TEMPTRANS := TEMPTRANS^.NEXTTRANS ;
              IF TEMPTRANS^.TRANSNAME = CHECKID THEN BEGIN
                  PREVIOUSTRANS^.NEXTTRANS :=
                         TEMPTRANS^.NEXTTRANS ;
                  WRITELN(' TRANSACTION DELETED- ',CHECKID) ;
                  OLDUSER^.TRANSCOUNT:=OLDUSER^.TRANSCOUNT - 1 ;
                  DISPOSE(TEMPTRANS) ;
                  FOUND := TRUE ; END
            END ; (* WHILE *)
         END ; (* ELSE *)
      END ; (* IF *)
END ; (* DELETETRANS *)
```

```
(************************************************************)
(*
(* PROCEDURE LISTCLUSTER DISPLAYS ON THE TERMINAL THE USER
(* IDENTIFIERS IN THE CURRENT CLUSTER GROUPING.
(* CALLING FORMAT : LISTCLUSTER ;
(*
(************************************************************)

PROCEDURE LISTCLUSTER ;
VAR
   INDEX : INTEGER ;
BEGIN
   INDEX := 1 ;
   WRITELN ;
   WRITELN(' FOLLOWING USERS HAVE COMMON TRANSACTIONS -') ;
   WHILE CLUSTER[INDEX] <> NIL DO BEGIN
      WRITE('  ',CLUSTER[INDEX]^.NAME) ;
      IF INDEX MOD 6 = 0 THEN WRITELN ;
      INDEX := INDEX + 1 ;
   END ;
END ;
```

```
(*****************************************************************)
(*
(* PROCEDURE BUILDCLUSTER IS DELETES FROM THE WORKING USER
(* DATABASE ALL USER THAT DO NOT HAVE SPECIFIC TRANSACTIONS
(* OR SYSTEM FUNCTIONS IN COMMON WITH OTHER USERS.  TERMINAL
(* INPUT SPECIFIES THE TRANSACTIONS/FUNCTIONS TO BE CHECKED
(* FOR.  THIS PROCESS CAN BE CONTINUED INDEFINITELY, WITH THE
(* USERS STILL IN THE DATABASE BEING DISPLAYED EACH TIME
(* NEW SPECIFICATIONS HAVE BEEN APPLIED.
(*
(*****************************************************************)

PROCEDURE BUILDCLUSTER ;
VAR
   OLDINDEX, NEWINDEX : INTEGER ;
   NEXTUSER : USERRECORD ;
BEGIN
   GETTRANSID(TRANSID) ;
   NEWINDEX := 1 ;
   NEXTUSER := USERHEAD ;
   WHILE NEXTUSER<>NIL DO BEGIN
      IF (DUPLICATETRANS(TRANSID,NEXTUSER)) THEN BEGIN
         CLUSTER[NEWINDEX] := NEXTUSER ;
         NEWINDEX := NEWINDEX + 1 ;
      END ;
      NEXTUSER := NEXTUSER^.NEXT ;
   END ;
   CLUSTER[NEWINDEX] := NIL ;
   GETTRANSID(TRANSID) ;
   WHILE TRANSID <> 'END       ' DO BEGIN
      NEWINDEX := 1 ;
      OLDINDEX := 1 ;
      NEXTUSER := CLUSTER[OLDINDEX] ;
      WHILE NEXTUSER <> NIL DO BEGIN
         IF (DUPLICATETRANS(TRANSID,NEXTUSER)) THEN BEGIN
            CLUSTER[NEWINDEX] := CLUSTER[OLDINDEX] ;
            NEWINDEX := NEWINDEX + 1 ;
         END ;
         OLDINDEX := OLDINDEX + 1 ;
         NEXTUSER := CLUSTER[OLDINDEX] ;
      END ;
      CLUSTER[NEWINDEX] := NIL ;
      LISTCLUSTER ;
      GETTRANSID(TRANSID) ;
   END ;
END ;
```

```
(**************************************************************)
(*
(*  PROCEDURE BUILDSUBCLUSTER IS REDUCES THE CURRENT WORK
(*  DATABASE TO ONLY ONE GROUP OF USER RECORDS BASED ON
(*  THE OFFICE IDENTIFIER INPUT BY THE USER.  THE DATABASE
(*  IS DESTROYED DURING THIS OPERATION; THEREFORE CARE MUST
(*  BE TAKEN TO PRECLUDE INADVERTENTLY DESTROYING A DATABASE
(*  COPY THAT HAS NOT BEEN PROTECTED PROPERLY.
(*  CALLING FORMAT:  BUILDSUBCLUSTER ;
(*
(**************************************************************)

PROCEDURE BUILDSUBCLUSTER ;
LABEL 2 ;
VAR
   TEMPUSER, LASTENTRY : USERRECORD ;
BEGIN
   GETOFFICEID(OFFICEID) ;
   IF OFFICEID = 'END       ' THEN GOTO 2 ;
   TEMPUSER := USERHEAD ;
   WHILE (TEMPUSER^.OFFICE <> OFFICEID) DO
      TEMPUSER := TEMPUSER^.NEXT ;
   USERHEAD := TEMPUSER ;
   TEMPUSER := TEMPUSER^.NEXT ;
   LASTENTRY := USERHEAD ;
   WHILE (TEMPUSER <> NIL) DO BEGIN
      IF TEMPUSER^.OFFICE = OFFICEID THEN BEGIN
         LASTENTRY^.NEXT := TEMPUSER ;
         LASTENTRY := TEMPUSER ;
         END ;
      TEMPUSER := TEMPUSER^.NEXT ;
   END ;
   LASTENTRY^.NEXT := NIL ;
2: END ;
```

```
BEGIN (* MAIN *)
   USERHEAD := NIL ;
10:
   GETCOMMAND(COMMAND) ;
   IF COMMAND = 'ADD USER   ' THEN BEGIN
20:   GETUSERID(USERID) ;
      IF USERID = 'END        ' THEN GOTO 10 ;
      IF DUPLICATEUSERID(USERID,USERPOINTER) THEN BEGIN
         WRITELN(' USER ALREADY EXISTS') ;
         GOTO 20 ;
        END ;
      BUILDUSER(USERID) ;
      GOTO 20 ;
     END
   ELSE BEGIN
      IF COMMAND = 'ADD TRANS ' THEN BEGIN
         GETUSERID(USERID);
         IF NOT DUPLICATEUSER(USERID,USERPOINTER) THEN BEGIN
            WRITELN(' USER DOES NOT EXIST ********** ');
            GOTO 10 ;
           END ;
80:
         GETTRANSID(TRANSID) ;
         IF TRANSID = 'END        ' THEN GOTO 10 ;
         IF (DUPLICATETRANS(TRANSID,USERPOINTER)) THEN BEGIN
            WRITELN(' USER ALREADY HAS THIS TRANSACTION') ;
            GOTO 80 ;
           END ;
         BUILDTRANS(TRANSID,USERPOINTER) ;
         GOTO 80 ;
        END
      ELSE BEGIN
         IF COMMAND = 'END        ' THEN
            GOTO 99
         ELSE BEGIN
            IF COMMAND = 'READ INPUT' THEN BEGIN
               READOLDFILE ;
               GOTO 10 ;
              END
            ELSE BEGIN
               IF COMMAND = 'SAVE DATA ' THEN BEGIN
                  SAVEDATA ;
                  GOTO 99 ;
                 END
               ELSE BEGIN
                  IF COMMAND = 'DEL TRANS ' THEN BEGIN
                     GETUSERID(USERID) ;
                     IF NOT DUPLICATEUSER(USERID,USERPOINTER)
                        THEN BEGIN
                        WRITELN(' USER DOES NOT EXIST ') ;
                        GOTO 10 ;
                       END ;
75:                  GETTRANSID(TRANSID) ;
                     IF TRANSID = 'END        ' THEN GOTO 10 ;
                     DELETETRANS(USERPOINTER,TRANSID) ;
                     GOTO 75 ;
                    END
```

```
                    ELSE BEGIN
                       IF COMMAND='DEL USER   ' THEN BEGIN
                          GETUSERID(USERID) ;
                          IF DUPLICATEUSER(USERID,USERPOINTER)
                             THEN BEGIN
                             WRITELN(' USER DELETED- ',USERID) ;
                             DELETEUSER(USERPOINTER) ;
                             GOTO 10 ;
                          END ;
                          WRITELN(' USER DOES NOT EXIST') ;
                       END
                       ELSE BEGIN
                          IF COMMAND = 'CLUSTER    ' THEN
                             BUILDCLUSTER
                          ELSE BEGIN
                             IF COMMAND='ADD OFFICE' THEN BEGIN
85:                             GETOFFICEID(OFFICEID) ;
                                IF NOT (OFFICEID='END        ')
                                   THEN BEGIN
                                   ADDOFFICE(OFFICEID) ;
                                   GOTO 85 ;
                                  END ;
                             END ;
                           IF COMMAND='SUBCLUSTER' THEN BEGIN
                              WRITELN(' COMMAND DESTROYS WORKING ',
                              'DATABASE. CONTINUE?(YES/NO)');
                              READLN ;
                              GETALFA(COMMAND) ;
                              IF COMMAND<>'YES        '
                                 THEN GOTO 10 ;
                              BUILDSUBCLUSTER ;
                              GOTO 10 ;
                             END ;
                         IF COMMAND='COMPARE    ' THEN BUILDMATRIX ;
                         IF COMMAND='SORT USERS' THEN SORTUSERS ;
                         END ;
                       END ;
                    GOTO 10 ;
                    END ;
                   END ; (* ELSE *)
                 END ;
               END ;
             END ;
          END ;
99:
END.
```

Appendix C

Distributed System Clusters

The system clusters are listed on the following pages in the similarity matrix format illustrated in Figure 2-5. The rows and columns are headed by the terminal identifier, with the computed percentage of similarity for two users being located at the intersection of the row of one, and the column of the other. At the bottom of each cluster matrix are the cluster name and the average cluster similarity value. The computation of this average value is described in the Relationship Evaluation section of Chapter IV.

Term	G13	G29	G416	G417	G418	G605	G60
G13		1.0	.62	.50	.70	.25	.67
G29	1.0		.62	.50	.70	.25	.67
G416	.62	.62		.50	.56	.17	.50
G417	.50	.50	.50		.30	.20	.57
G418	.70	.70	.56	.30		.22	.60
G605	.25	.25	.17	.20	.22		.12
G60	.67	.67	.50	.57	.60	.12	

Headquarters ESC, Kelly AFB Cluster
Average Similarity 48.7%

Term	F17	G200	G211	G40	G603
F17		1.0	1.0	1.0	.67
G200	1.0		1.0	1.0	.67
G211	1.0	1.0		1.0	.67
G40	1.0	1.0	1.0		.67
G603	.67	.67	.67	.67	

Chanute Technical Training Center Cluster
Average Similarity 86.8%

Term	G388	G389	G390
G388		.70	.40
G389	.70		.62
G390	.42	.62	

Headquarters Alaskan Air Command Cluster
Average Similarity 57.3%

Term	B57	F13	G207	G208	G209	G210	G604
B57		1.0	.50	.50	.75	1.0	.67
F13	1.0		.50	.50	.75	1.0	.67
G207	.50	.50		1.0	.40	.50	.67
G208	.50	.50	1.0		.40	.50	.67
G209	.75	.75	.40	.40		.75	.50
G210	1.0	1.0	.50	.50	.75		.67
G604	.67	.67	.67	.67	.50	.67	

Sheppard Technical Training Center Cluster
Average Similarity 66.2%

Term	B44	G201	G204	G213	G57	G602
B44		1.0	1.0	1.0	1.0	1.0
G201	1.0		1.0	1.0	1.0	1.0
G204	1.0	1.0		1.0	1.0	1.0
G213	1.0	1.0	1.0		1.0	1.0
G57	1.0	1.0	1.0	1.0		1.0
G602	1.0	1.0	1.0	1.0	1.0	

Keesler Technical Training Center Cluster
Average Similarity 100 %

Term	F11	F2	G373	G374	G375	G39	G510	G63
F11		.44	.40	.44	.50	.25	.37	.50
F2	.44		.70	.78	.67	.44	.75	.67
G373	.40	.70		.89	.78	.56	.50	.78
G374	.44	.78	.89		.87	.62	.56	.87
G375	.50	.67	.78	.87		.71	.62	.75
G39	.25	.44	.56	.62	.71		.37	.50
G510	.37	.75	.50	.56	.62	.37		.44
G63	.50	.67	.78	.87	.75	.50	.44	

Headquarters USAFE Cluster
Average Similarity 59.8%

Term	G376	G377	G378	G379	G413	G414
G376		.62	.83	.62	.67	.62
G377	.62		.50	.75	.45	1.0
G378	.83	.50		.50	.56	.50
G379	.62	.75	.50		.60	.75
G413	.67	.45	.56	.60		.45
G414	.62	1.0	.50	.75	.45	

Headquarters PACAF Cluster
Average Similarity 62.8%

Term	B109	B79	B95	F16	F1	G311	G392	G50	G53
B109		.37	.20	.50	.40	.57	.50	.33	.50
B79	.37		.00	.62	.67	.50	.45	.50	.62
B95	.20	.00		.00	.00	.17	.10	.00	.00
F16	.50	.62	.00		.60	.62	.55	.43	.56
F1	.40	.67	.00	.60		.50	.46	.33	.78
G311	.57	.50	.17	.62	.50		.60	.50	.44
G392	.50	.45	.10	.55	.46	.60		.30	.42
G50	.33	.50	.00	.43	.33	.50	.30		.43
G53	.50	.62	.00	.56	.78	.44	.42	.43	

Bolling AFB Cluster
Average Similarity 40.3%

AD-A115 533 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
A DISTRIBUTED PROCESSING DESIGN FOR THE PERSONNEL DATA SYSTEM'S--ETC(U)
DEC 81 R V BRANDT
UNCLASSIFIED AFIT/GCS/EE/81D-4 NL

2 OF 2
AD A115533

END
DATE
FILMED
7-82
DTIC

Term	B111	G126	G28	G380	G381	G382	G383	G384	G385	G386	G387	G505
B111		.17	.25	.43	.50	.37	.50	.27	.30	.33	.57	.29
G126	.17		.40	.40	.20	.14	.50	.22	.25	.29	.14	.50
G28	.25	.40		.43	.50	.37	.50	.56	.44	.50	.57	.80
G380	.43	.40	.43		.50	.57	.80	.56	.62	.50	.37	.50
G381	.50	.20	.50	.50		.43	.60	.44	.50	.57	.67	.60
G382	.37	.14	.37	.57	.43		.43	.67	.40	.30	.50	.25
G383	.50	.50	.50	.80	.60	.43		.44	.50	.57	.43	.60
G384	.27	.22	.56	.56	.44	.67	.44		.70	.60	.50	.44
G385	.30	.25	.44	.62	.50	.40	.50	.70		.87	.40	.50
G386	.33	.29	.50	.50	.57	.30	.57	.60	.87		.44	.57
G387	.57	.14	.57	.37	.67	.50	.43	.50	.40	.44		.43
G505	.29	.50	.80	.50	.60	.25	.60	.44	.50	.57	.43	

Andrews AFB Cluster
Average Similarity 46.2%

Term	G302	G304	G330	G331	G332	G333	G334	G335	G500	G504	G91
G302		.50	.30	.30	.37	.33	.44	.44	.44	.11	.00
G304	.50		.57	.57	.50	.56	.57	.57	.57	.33	.00
G330	.30	.57		1.0	.43	.50	.50	.50	.50	.50	.00
G331	.30	.57	1.0		.43	.50	.50	.50	.50	.50	.00
G332	.37	.50	.43	.43		.30	.67	.67	.67	.40	.00
G333	.33	.56	.50	.50	.30		.36	.36	.36	.33	.00
G334	.44	.57	.50	.50	.67	.36		1.0	1.0	.29	.00
G335	.44	.57	.50	.50	.67	.36	1.0		1.0	.29	.00
G500	.44	.57	.50	.50	.67	.36	1.0	1.0		.29	.00
G504	.11	.33	.50	.50	.40	.33	.29	.29	.29		.00
G91	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	

Headquarters AFRES Cluster
Average Similarity 46.2%

Term	B103	B104	B81	G107	G25	G2	G30	G31	G32	G35	G71
B103		1.0	.67	.83	1.0	.83	1.0	.83	.83	.83	.67
B104	1.0		.67	.83	1.0	.83	1.0	.83	.83	.83	.67
B81	.67	.67		.80	.67	.80	.67	.80	.80	.80	1.0
G107	.83	.83	.80		.83	1.0	.83	1.0	1.0	1.0	.80
G25	1.0	1.0	.67	.83		.83	1.0	.83	.83	.83	.67
G2	.83	.83	.80	1.0	.83		.83	1.0	1.0	1.0	.80
G30	1.0	1.0	.67	.83	1.0	.83		.83	.83	.83	.67
G31	.83	.83	.80	1.0	.83	1.0	.83		1.0	1.0	.80
G32	.83	.83	.80	1.0	.83	1.0	.83	1.0		1.0	.80
G35	.83	.83	.80	1.0	.83	1.0	.83	1.0	1.0		.80
G71	.67	.67	1.0	.80	.67	.80	.67	.80	.80	.80	

Office of Civilian Personnel Operations Cluster
Average Similarity 86.8%

Term	B77	B78	F108	F111	G120	G121	G127	G203	G55	G56
B77		.75	.25	.00	.50	.25	.20	1.0	.50	.00
B78	.75		.20	.00	.40	.20	.40	.75	.40	.00
F108	.25	.20		.50	.25	.33	.00	.25	.25	.00
F111	.00	.00	.50		.33	.50	.00	.00	.33	.00
G120	.50	.40	.25	.33		.67	.20	.50	1.0	.00
G121	.25	.20	.33	.50	.67		.25	.25	.67	.00
G127	.20	.40	.00	.00	.20	.25		.20	.20	.00
G203	1.0	.75	.25	.00	.50	.25	.20		.50	.00
G55	.50	.40	.25	.33	1.0	.67	.20	.50		.00
G56	.00	.00	.00	.00	.00	.00	.00	.00	.00	

Lackland AFB Cluster
Average Similarity 36.1%

Term	B105	B106	B110	B66	B88	FA0	G102	G391	G84
B105		.37	.56	.62	.45	.50	.50	.50	.62
B106	.37		.37	.67	.44	.50	.50	.50	.43
B110	.56	.37		.62	.45	.50	.50	.50	.62
B66	.62	.67	.62		.67	.57	.57	.57	.71
B88	.45	.44	.45	.67		.56	.56	.56	.50
FA0	.50	.50	.50	.57	.56		.67	.67	.57
G102	.50	.50	.50	.57	.56	.67		.67	.57
G391	.50	.50	.50	.57	.56	.67	.67		.50
G84	.62	.43	.62	.71	.50	.57	.57	.50	

Pentagon Cluster # 1
Average Similarity 54.8%

Term	B102	B107	B114	B68	B86	B91	G393	G394	G399
B102		.12	.14	.09	.09	.09	.09	.14	.14
B107	.12		.12	.25	.25	.25	.25	.12	.12
B114	.14	.12		.33	.50	.33	.50	.60	.78
B68	.09	.25	.33		.33	.33	.60	.33	.33
B86	.09	.25	.50	.33		.60	.60	.50	.50
B91	.09	.25	.33	.33	.60		.60	.33	.33
G393	.09	.25	.50	.60	.60	.60		.50	.50
G394	.14	.12	.60	.33	.50	.33	.50		.60
G399	.14	.12	.78	.33	.50	.33	.50	.60	

Pentagon Cluster # 2
Average Similarity 34.1%

Term	G319	G324	G325	G326	G327	G329	G64
G319		.67	.67	.57	.43	.57	.29
G324	.67		.60	.80	.60	.80	.40
G325	.67	.60		.50	.60	.50	.40
G326	.57	.80	.50		.80	1.0	.60
G327	.43	.60	.60	.80		.80	.75
G329	.57	.80	.50	1.0	.80		.60
G64	.29	.40	.40	.60	.75	.60	

Wright-Patterson AFB Cluster # 1
Average Similarity 61.4%

Term	G305	G320	G321	G322	G323	G328	G78
G305		.78	.67	.44	.50	.25	1.0
G320	.78		.87	.62	.67	.25	.78
G321	.67	.87		.50	.75	.29	.67
G322	.44	.62	.50		.50	.40	.44
G323	.50	.67	.75	.50		.29	.50
G328	.25	.25	.29	.40	.29		.25
G78	1.0	.78	.67	.44	.50	.25	

Wright-Patterson AFB Cluster # 2
Average Similarity 54.4%

Term	G314	G364	G365	G366	G367	G368	G370	G70
G314		.73	.89	.73	.80	.73	.73	.78
G364	.73		.80	1.0	.90	1.0	.82	.70
G365	.89	.80		.80	.89	.80	.80	.87
G366	.73	1.0	.80		.90	1.0	.82	.70
G367	.80	.90	.89	.90		.90	.90	.78
G368	.73	1.0	.80	1.0	.90		.82	.70
G370	.73	.82	.80	.82	.90	.82		.70
G70	.78	.70	.87	.70	.78	.70	.70	

Langley AFB Cluster # 1
Average Similarity 82.1%

Term	G306	G369	G371	G372	G396	G508	G509	G511
G306		.33	.17	.17	.25	.33	.17	.50
G369	.33		.83	.57	.75	.33	.57	.17
G371	.17	.83		.67	.62	.40	.67	.20
G372	.17	.57	.67		.62	.40	.67	.20
G396	.25	.75	.62	.62		.25	.62	.12
G508	.33	.33	.40	.40	.25		.40	.50
G509	.17	.57	.67	.67	.62	.40		.20
G511	.50	.17	.20	.20	.12	.50	.20	

Langley AFB Cluster # 2
Average Similarity 41.7%

Term	G316	G351	G352	G354	G355	G357	G358	G404	G405	G507
G316		.86	.75	.57	.62	.62	.71	.67	.56	.67
G351	.86		.87	.50	.56	.56	.62	.78	.67	.57
G352	.75	.87		.62	.67	.67	.75	.89	.60	.50
G354	.57	.50	.62		.71	.71	.83	.56	.44	.50
G355	.62	.56	.67	.71		.75	.86	.60	.50	.57
G357	.62	.56	.67	.71	.75		.86	.60	.50	.57
G358	.71	.62	.75	.83	.86	.86		.67	.56	.67
G404	.67	.78	.89	.56	.60	.60	.67		.70	.44
G405	.56	.67	.60	.44	.50	.50	.56	.70		.50
G507	.67	.57	.50	.50	.57	.57	.67	.44	.50	

Scott AFB Cluster # 1
Average Similarity 64.4%

Term	G353	G356	G359	G360	G361	G362	G363	G406	G407
G353		.37	.40	.50	.44	.44	.44	.50	.56
G356	.37		.50	.62	.56	.56	.56	.62	.50
G359	.40	.50		.78	.89	.89	.89	.78	.64
G360	.50	.62	.78		.87	.87	.87	1.0	.78
G361	.44	.56	.89	.87		1.0	.78	.87	.70
G362	.44	.56	.89	.87	1.0		.78	.87	.70
G363	.44	.56	.89	.87	.78	.78		.87	.70
G406	.50	.62	.78	1.0	.87	.87	.87		.78
G407	.56	.50	.64	.78	.70	.70	.70	.78	

Scott AFB Cluster # 2
Average Similarity 69.2%

Term	G339	G340	G341	G343	G346	G347	G349	G350	G401	G403
G339		.44	.75	.87	.78	.62	.44	.44	.62	.78
G340	.44		.44	.40	.50	.50	.71	1.0	.71	.67
G341	.75	.44		.87	.60	.44	.44	.44	.44	.60
G343	.87	.40	.87		.70	.56	.40	.40	.56	.70
G346	.78	.50	.60	.70		.50	.67	.50	.67	.80
G347	.62	.50	.44	.56	.50		.50	.50	.71	.50
G349	.44	.71	.44	.40	.67	.50		.71	.71	.50
G350	.44	1.0	.44	.40	.50	.50	.71		.71	.67
G401	.62	.71	.44	.56	.67	.71	.71	.71		.67
G403	.78	.67	.60	.70	.80	.50	.50	.67	.67	

Headquarters SAC Cluster # 1
Average Similarity 60.4%

Term	G301	G338	G342	G344	G345	G348	G400	G402	G506
G301		.50	.43	.57	.62	.57	.50	.43	.50
G338	.50		.71	.86	.67	.86	.56	.71	.57
G342	.43	.71		.83	.62	.83	.50	1.0	.50
G344	.57	.86	.83		.75	1.0	.62	.83	.67
G345	.62	.67	.62	.75		.75	.67	.62	.50
G348	.57	.86	.83	1.0	.75		.62	.83	.67
G400	.50	.56	.50	.62	.67	.62		.50	.37
G402	.43	.71	1.0	.83	.62	.83	.50		.50
G506	.50	.57	.50	.67	.50	.67	.37	.50	

Headquarters SAC Cluster # 2
Average Similarity 64.6%

Term	B54	B55	B56	B58	B59	G125	G410	G412	G600	G85
B54		.43	.67	.67	.40	.50	.80	.80	.40	.20
B55	.43		.67	.67	.40	.50	.50	.50	.40	.20
B56	.67	.67		1.0	.40	.80	.80	.80	.40	.20
B58	.67	.67	1.0		.40	.80	.80	.80	.40	.20
B59	.40	.40	.40	.40		.50	.50	.50	1.0	.50
G125	.50	.50	.80	.80	.50		.60	.60	.50	.25
G410	.80	.50	.80	.80	.50	.60		1.0	.50	.25
G412	.80	.50	.80	.80	.50	.60	1.0		.50	.25
G600	.40	.40	.40	.40	1.0	.50	.50	.50		.50
G85	.20	.20	.20	.20	.50	.25	.25	.25	.50	

Headquarters ATC Cluster # 1
Average Similarity 54.4%

Term	G122	G123	G21	G22	G23	G33	G34	G408	G409	G411
G122		.14	.25	.33	.50	.22	.50	.22	.50	.44
G123	.14		.33	.43	.25	.29	.25	.29	.25	.22
G21	.25	.33		.50	.50	.83	.50	.57	.50	.44
G22	.33	.43	.50		.40	.44	.56	.44	.40	.50
G23	.50	.25	.50	.40		.44	.75	.44	1.0	.87
G33	.22	.29	.83	.44	.44		.44	.71	.44	.40
G34	.50	.25	.50	.56	.75	.44		.44	.75	.87
G408	.22	.29	.57	.44	.44	.71	.44		.44	.40
G409	.50	.25	.50	.40	1.0	.44	.75	.44		.87
G411	.44	.22	.44	.50	.87	.40	.87	.40	.87	

Headquarters ATC Cluster # 2
Average Similarity 47.2%

Term	B52	B53	F101	G52	G702	G703	G704	G705	G706
B52		1.0	.50	.30	.00	.00	.00	.00	.00
B53	1.0		.50	.30	.00	.00	.00	.00	.00
F101	.50	.50		.56	.00	.00	.00	.00	.00
G52	.30	.30	.56		.00	.00	.00	.00	.00
G702	.00	.00	.00	.00		.00	.00	.00	.00
G703	.00	.00	.00	.00	.00		.00	.00	.00
G704	.00	.00	.00	.00	.00	.00		.00	.00
G705	.00	.00	.00	.00	.00	.00	.00		.00
G706	.00	.00	.00	.00	.00	.00	.00	.00	

Headquarters ATC Cluster # 3
Average Similarity 54.3%

Term	B64	B65	G109	G251	G253	G254	G257	G259	G46	G47
B64		1.0	.82	.70	.58	.90	.60	.60	.80	.60
B65	1.0		.82	.70	.58	.90	.60	.60	.80	.60
G109	.82	.82		.55	.58	.73	.45	.45	.80	.45
G251	.70	.70	.55		.60	.78	.62	.62	.50	.86
G253	.58	.58	.58	.60		.64	.50	.50	.55	.67
G254	.90	.90	.73	.78	.64		.67	.67	.70	.67
G257	.60	.60	.45	.62	.50	.67		1.0	.56	.71
G259	.60	.60	.45	.62	.50	.67	1.0		.56	.71
G46	.80	.80	.80	.50	.55	.70	.56	.56		.56
G47	.60	.60	.45	.86	.67	.67	.71	.71	.56	

Lowry AFB Cluster # 1
Average Similarity 64.1%

Term	B62	G202	G212	G252	G256	G258	G317	G318	G4	G58	G601
B62		.20	.22	.67	.78	.78	.44	.50	.22	.22	.11
G202	.20		.75	.22	.33	.33	.50	.33	.75	.75	.50
G212	.22	.75		.25	.37	.37	.60	.40	1.0	1.0	.67
G252	.67	.22	.25		.67	.67	.50	.57	.25	.25	.12
G256	.78	.33	.37	.67		1.0	.62	.50	.37	.37	.25
G258	.78	.33	.37	.67	1.0		.62	.50	.37	.37	.25
G317	.44	.50	.60	.50	.62	.62		.50	.60	.60	.40
G318	.50	.33	.40	.57	.50	.50	.50		.40	.40	.20
G4	.22	.75	1.0	.25	.37	.37	.60	.40		1.0	.67
G584	.22	.75	1.0	.25	.37	.37	.60	.40	1.0		.67
G601	.11	.50	.67	.12	.25	.25	.40	.20	.67	.67	

Lowry AFB Cluster # 2
Average Similarity 49.0%

Term	B60	B61	B63	G108	G110	G111	G250	G255	G44	G45
B60		.64	.67	.40	.67	.40	.80	.70	.33	.38
B61	.64		.57	.23	.69	.33	.67	.90	.27	.43
B63	.67	.57		.50	.50	.38	.69	.50	.23	.57
G108	.40	.23	.50		.29	.50	.33	.25	.25	.45
G110	.67	.69	.50	.29		.38	.57	.75	.23	.29
G111	.40	.33	.38	.50	.38		.33	.36	.25	.45
G250	.80	.67	.69	.33	.57	.33		.58	.27	.43
G255	.70	.90	.50	.25	.75	.36	.58		.30	.36
G44	.33	.27	.23	.25	.23	.25	.27	.30		.40
G45	.38	.43	.57	.45	.29	.45	.43	.36	.40	

Lowry AFB Cluster # 3
Average Similarity 45.5%

Term	B11	B12	B16	B17	B18	B20	B41	B69	G8	G99
B11		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B12	1.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B16	1.0	1.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0
B17	1.0	1.0	1.0		1.0	1.0	1.0	1.0	1.0	1.0
B18	1.0	1.0	1.0	1.0		1.0	1.0	1.0	1.0	1.0
B20	1.0	1.0	1.0	1.0	1.0		1.0	1.0	1.0	1.0
B41	1.0	1.0	1.0	1.0	1.0	1.0		1.0	1.0	1.0
B69	1.0	1.0	1.0	1.0	1.0	1.0	1.0		1.0	1.0
G8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0		1.0
G99	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	

AFMPC Cluster # 1
Average Similarity 100%

Term	B33	B35	B37	B40	B46	B47	B49	B50	B51	B76	G12	G19
B33		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B35	1.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B37	1.0	1.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B40	1.0	1.0	1.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B46	1.0	1.0	1.0	1.0		1.0	1.0	1.0	1.0	1.0	1.0	1.0
B47	1.0	1.0	1.0	1.0	1.0		1.0	1.0	1.0	1.0	1.0	1.0
B49	1.0	1.0	1.0	1.0	1.0	1.0		1.0	1.0	1.0	1.0	1.0
B50	1.0	1.0	1.0	1.0	1.0	1.0	1.0		1.0	1.0	1.0	1.0
B51	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0		1.0	1.0	1.0
B76	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0		1.0	1.0
G12	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0		1.0
G19	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	

AFMPC Cluster # 2
Average Similarity 100 %

Term	FA2	F5	F6	F8	G81	G82	G83	G89	G92	G93
FA2		1.0	1.0	.80	1.0	1.0	1.0	.80	1.0	.80
F5	1.0		1.0	.80	1.0	1.0	1.0	.80	1.0	.80
F6	1.0	1.0		.80	1.0	1.0	1.0	.80	1.0	.80
F8	.80	.80	.80		.80	.80	.80	1.0	.80	1.0
G81	1.0	1.0	1.0	.80		1.0	1.0	.80	1.0	.80
G82	1.0	1.0	1.0	.80	1.0		1.0	.80	1.0	.80
G83	1.0	1.0	1.0	.80	1.0	1.0		.80	1.0	.80
G89	.80	.80	.80	1.0	.80	.80	.80		.80	1.0
G92	1.0	1.0	1.0	.80	1.0	1.0	1.0	.80		.80
G93	.80	.80	.80	1.0	.80	.80	.80	1.0	.80	

AFMPC Cluster # 3
Average Similarity 90.6%

Term	B100	B36	B45	B48	B67	B82	B87	G10	G74	G76
B100		.78	.89	.89	.78	.78	.89	.78	.89	1.0
B36	.78		.89	.89	1.0	1.0	.89	1.0	.89	.78
B45	.89	.89		1.0	.89	.89	1.0	.89	1.0	.89
B48	.89	.89	1.0		.89	.89	1.0	.89	1.0	.89
B67	.78	1.0	.89	.89		1.0	.89	1.0	.89	.78
B82	.78	1.0	.89	.89	1.0		.89	1.0	.89	.78
B87	.89	.89	1.0	1.0	.89	.89		.89	1.0	.89
G10	.78	1.0	.89	.89	1.0	1.0	.89		.89	.78
G74	.89	.89	1.0	1.0	.89	.89	1.0	.89		.89
G76	1.0	.78	.89	.89	.78	.78	.89	.78	.89	

AFMPC Cluster # 4
Average Similarity 90.2%

Term	B13	B15	B26	B28	B32	B71	B80	G18	G42	G73
B13		.90	.80	.80	1.0	.90	1.0	1.0	.90	.90
B15	.90		.89	.89	.90	1.0	.90	.90	1.0	1.0
B26	.80	.89		1.0	.80	.89	.80	.80	.89	.89
B28	.80	.89	1.0		.80	.89	.80	.80	.89	.89
B32	1.0	.90	.80	.80		.90	1.0	1.0	.90	.90
B71	.90	1.0	.89	.89	.90		.90	.90	1.0	1.0
B80	1.0	.90	.80	.80	1.0	.90		1.0	.90	.90
G18	1.0	.90	.80	.80	1.0	.90	1.0		.90	.90
G42	.90	1.0	.89	.89	.90	1.0	.90	.90		1.0
G73	.90	1.0	.89	.89	.90	1.0	.90	.90	1.0	

AFMPC Cluster # 5
Average Similarity 90.9%

Term	B22	B23	B25	B75	B7	B84	B85	G11	G41	G43	G49	G75
B22		1.0	1.0	.78	.80	1.0	.89	.73	.89	.73	.73	.73
B23	1.0		1.0	.78	.80	1.0	.89	.73	.89	.73	.73	.73
B25	1.0	1.0		.78	.80	1.0	.89	.73	.89	.73	.73	.73
B75	.78	.78	.78		.80	.78	.89	.73	.89	.73	.73	.73
B7	.80	.80	.80	.80		.80	.90	.75	.90	.75	.75	.75
B84	1.0	1.0	1.0	.78	.80		.89	.73	.89	.73	.73	.73
B85	.89	.89	.89	.89	.90	.89		.82	1.0	.82	.82	.82
G11	.73	.73	.73	.73	.75	.73	.82		.82	1.0	.83	.83
G41	.89	.89	.89	.89	.90	.89	1.0	.82		.82	.82	.82
G43	.73	.73	.73	.73	.75	.73	.82	1.0	.82		.83	.83
G49	.73	.73	.73	.73	.75	.73	.82	.83	.82	.83		.83
G75	.73	.73	.73	.73	.75	.73	.82	.83	.82	.83	.83	

AFMPC Cluster # 6
Average Similarity 81.9%

Term	B14	B19	B21	B24	B27	B38	B42	B43	B70	B74
B14		.80	1.0	.78	.60	.50	.64	.50	1.0	.67
B19	.80		.80	.60	.60	.64	.50	.64	.80	.67
B21	1.0	.80		.78	.60	.50	.64	.50	1.0	.67
B24	.78	.60	.78		.75	.60	.78	.60	.78	.86
B27	.60	.60	.60	.75		.60	.60	.60	.60	.62
B38	.50	.64	.50	.60	.60		.80	1.0	.50	.67
B42	.64	.50	.64	.78	.60	.80		.80	.64	.67
B43	.50	.64	.50	.60	.60	1.0	.80		.50	.67
B70	1.0	.80	1.0	.78	.60	.50	.64	.50		.67
B74	.67	.67	.67	.86	.62	.67	.67	.67	.67	

AFMPC Cluster # 7
Average Similarity 68.3%

Term	B113	B29	B93	B99	FA14	FA15	FA16	FA17	G16	G5
B113		1.0	.67	.57	.67	.67	.50	.83	1.0	.57
B29	1.0		.67	.57	.67	.67	.50	.83	1.0	.57
B93	.67	.67		.80	1.0	1.0	.75	.80	.67	.80
B99	.57	.57	.80		.80	.80	.60	.67	.57	1.0
FA14	.67	.67	1.0	.80		1.0	.75	.80	.67	.80
FA15	.67	.67	1.0	.80	1.0		.75	.80	.67	.80
FA16	.50	.50	.75	.60	.75	.75		.60	.50	.60
FA17	.83	.83	.80	.67	.80	.80	.60		.83	.67
G16	1.0	1.0	.67	.57	.67	.67	.50	.83		.57
G5	.57	.57	.80	1.0	.80	.80	.60	.67	.57	

AFMPC Cluster # 8
Average Similarity 73.4%

Term	B2	FA3	F179	F180	F7	G100	G20	G37	G62	G9
B2		.71	.62	.50	.75	.71	1.0	.71	.86	.62
FA3	.71		.83	.67	.71	1.0	.71	1.0	.83	.83
F179	.62	.83		.83	.86	.83	.62	.83	.71	.71
F180	.50	.67	.83		.71	.67	.50	.67	.57	.57
F7	.75	.71	.86	.71		.71	.75	.71	.62	.62
G100	.71	1.0	.83	.67	.71		.71	1.0	.83	.83
G20	1.0	.71	.62	.50	.75	.71		.71	.86	.62
G37	.71	1.0	.83	.67	.71	1.0	.71		.83	.83
G62	.86	.83	.71	.57	.62	.83	.86	.83		.71
G9	.62	.83	.71	.57	.62	.83	.62	.83	.71	

AFMPC Cluster # 9
Average Similarity 74.4%

Term	B112	B1	B30	B5	G114	G421	G67	G77	G7
B112		.87	1.0	.75	.60	.62	.87	.78	.87
B1	.87		.87	.67	.70	.75	.78	.89	1.0
B30	1.0	.87		.75	.60	.62	.87	.78	.87
B5	.75	.67	.75		.60	.62	.67	.78	.67
G114	.60	.70	.60	.60		.67	.55	.64	.70
G421	.62	.75	.62	.62	.67		.56	.67	.75
G67	.87	.78	.87	.67	.55	.56		.70	.78
G77	.78	.89	.78	.78	.64	.67	.70		.89
G7	.87	1.0	.87	.67	.70	.75	.78	.89	

AFMPC Cluster # 10
Average Similarity 74.3%

Term	FA10	F12	F181	F3	F4	F9	G101	G112	G1	G38
FA10		.62	.70	.62	.60	.67	.62	.75	.67	.67
F12	.62		.56	.67	.62	.71	.67	.83	.71	.71
F181	.70	.56		.56	.89	.78	.56	.67	.60	.78
F3	.62	.67	.56		.62	.71	.67	.83	.71	.71
F4	.60	.62	.89	.62		.87	.62	.75	.67	.87
F9	.67	.71	.78	.71	.87		.71	.86	.75	1.0
G101	.62	.67	.56	.67	.62	.71		.83	.71	.71
G112	.75	.83	.67	.83	.75	.86	.83		.86	.86
G1	.67	.71	.60	.71	.67	.75	.71	.86		.75
G38	.67	.71	.78	.71	.87	1.0	.71	.86	.75	

AFMPC Cluster # 11
Average Similarity 71.8%

Term	B116	B4	B6	B83	B92	B97	G14	G26	G79	G80	G87
B116		.60	.78	.75	.67	.56	.56	.67	.55	.75	.67
B4	.60		.64	.60	.70	.60	.60	.70	.73	.78	.55
B6	.78	.64		.60	.55	.60	.78	.70	.58	.78	.70
B83	.75	.60	.60		.67	.56	.56	.67	.55	.75	.67
B92	.67	.70	.55	.67		.50	.50	.60	.64	.67	.60
B97	.56	.60	.60	.56	.50		.75	.50	.55	.75	.50
G14	.56	.60	.78	.56	.50	.75		.67	.55	.75	.67
G26	.67	.70	.70	.67	.60	.50	.67		.64	.67	.78
G79	.55	.73	.58	.55	.64	.55	.55	.64		.70	.50
G80	.75	.78	.78	.75	.67	.75	.75	.67	.70		.67
G87	.67	.55	.70	.67	.60	.50	.67	.78	.50	.67	

AFMPC Cluster # 12
Average Similarity 64.3%

Term	B117	B8	FA12	FA4	F10	G106	G300	G6	G86	G95
B117		1.0	.25	.25	.50	.20	.67	.33	.33	.33
B8	1.0		.25	.25	.50	.20	.67	.33	.33	.33
FA12	.25	.25		.50	.40	.40	.20	.50	.50	.50
FA4	.25	.25	.50		.75	.75	.50	.50	.50	.50
F10	.50	.50	.40	.75		.60	.75	.67	.67	.67
G106	.20	.20	.40	.75	.60		.40	.43	.67	.43
G300	.67	.67	.20	.50	.75	.40		.50	.50	.50
G6	.33	.33	.50	.50	.67	.43	.50		.71	1.0
G86	.33	.33	.50	.50	.67	.67	.50	.71		.71
G95	.33	.33	.50	.50	.67	.43	.50	1.0	.71	

AFMPC Cluster # 13
Average Similarity 49.8%

Term	F106	F14	G103	G309	G512	G51	G61	G65	G88	G90
F106		.23	.23	.33	.09	.46	.36	.42	.46	.18
F14	.23		.43	.11	.00	.30	.62	.37	.44	.17
G103	.23	.43		.25	.20	.44	.44	.37	.30	.40
G309	.33	.11	.25		.20	.44	.18	.37	.30	.17
G512	.09	.00	.20	.20		.12	.12	.17	.00	.50
G51	.46	.30	.44	.44	.12		.45	.40	.60	.11
G61	.36	.62	.44	.18	.12	.45		.40	.60	.25
G65	.42	.37	.37	.37	.17	.40	.40		.40	.33
G88	.46	.44	.30	.30	.00	.60	.60	.40		.11
G90	.18	.17	.40	.17	.50	.11	.25	.33	.11	

AFMPC Cluster # 14
Average Similarity 30.7%

Term	B31	B34	G113	G119	G15	G17	G36	G54	G69	G96
B31		.33	.43	.40	.25	.50	.33	.50	.50	.80
B34	.33		.40	.37	.56	.62	.33	.44	.62	.44
G113	.43	.40		.50	.33	.57	.67	.37	.57	.57
G119	.40	.37	.50		.29	.60	.75	.33	.60	.60
G15	.25	.56	.33	.29		.57	.25	.37	.57	.37
G17	.50	.62	.57	.60	.57		.50	.67	1.0	.67
G36	.33	.33	.67	.75	.25	.50		.29	.50	.50
G54	.50	.44	.37	.33	.37	.67	.29		.67	.43
G69	.50	.62	.57	.60	.57	1.0	.50	.67		.67
G96	.80	.44	.57	.60	.37	.67	.50	.43	.67	

AFMPC Cluster # 15
Average Similarity 49.9%

Term	B10	B3	B73	B96	FA1	FA7	G115	G24	G27	G59	G72
B10		.09	.00	.08	.25	.00	.00	.00	.00	.00	.00
B3	.09		.36	.64	.25	.00	.36	.36	.15	.31	.18
B73	.00	.36		.23	.33	.00	.33	.33	.33	.43	.20
B96	.08	.64	.23		.23	.00	.33	.33	.14	.20	.17
FA1	.25	.25	.33	.23		.00	.33	.33	.33	.43	.20
FA7	.00	.00	.00	.00	.00		.00	.00	.00	.00	.00
G115	.00	.36	.33	.33	.33	.00		.33	.33	.56	.25
G24	.00	.36	.33	.33	.33	.00	.33		.14	.43	.50
G27	.00	.15	.33	.14	.33	.00	.33	.14		.25	.20
G59	.00	.31	.43	.20	.43	.00	.56	.43	.25		.33
G72	.00	.18	.20	.17	.20	.00	.25	.50	.20	.33	

AFMPC Cluster # 16
Average Similarity 25.7%

Term	B115	G205	G206	G395	G422	G502	G503	G68	G97
B115		1.0	1.0	.50	.33	.43	.33	.00	.67
G205	1.0		1.0	.50	.33	.43	.33	.00	.67
G206	1.0	1.0		.50	.33	.43	.33	.00	.67
G395	.50	.50	.50		.57	.62	.57	.00	.33
G422	.33	.33	.33	.57		.50	.43	.00	.17
G502	.43	.43	.43	.62	.50		.71	.00	.29
G503	.33	.33	.33	.57	.43	.71		.00	.17
G68	.00	.00	.00	.00	.00	.00	.00		.00
G97	.67	.67	.67	.33	.17	.29	.17	.00	

Randolph AFB Central Cluster # 1
Average Similarity 50.5%

Term	B108	G312	G313	G336	G337	G397	G398	G415	G501
B108		.80	.67	.56	.60	.37	.62	.60	.56
G312	.80		.70	.45	.80	.30	.50	.64	.45
G313	.67	.70		.62	.67	.43	.71	.67	.62
G336	.56	.45	.62		.56	.50	.57	.56	.71
G337	.60	.80	.67	.56		.37	.44	.60	.40
G397	.37	.30	.43	.50	.37		.60	.37	.29
G398	.62	.50	.71	.57	.44	.60		.62	.57
G415	.60	.64	.67	.56	.60	.37	.62		.56
G501	.56	.45	.62	.71	.40	.29	.57	.56	

Randolph AFB Central Cluster # 2
Average Similarity 55.7%

Appendix D

Software User's Guide

Introduction

This software package (Appendix B) supports the actions necessary to create and maintain a data base of doubly linked records, and the analysis of those records for the generation of clusters. As implemented for this thesis, the database consists of user records and transaction records. The clustering and comparison capabilities described later support generation of similarity matrices based on the percentage of transaction records which users in the "working database" have in common.

The software is written in Pascal for the Cyber computer and functions in an interactive mode with the user. To support large scale efforts with a database being created during several sessions, the package provides the capability to save the current working database and to subsequently use it as initial input for the addition of more records.

Concept of Operation

This package has been developed with minimal internal data editing to provide more generalized capabilities. Therefore, although this user's guide will continue to refer to the data items, variables, and records as they were implemented, the reader should primarily concentrate on the capabilities

provided rather than the implementation specific aspects.

User interaction is controlled through the main program command menu consisting of the twelve actions supported. Each of these menu items is described in the following paragraphs. Where some logical progression of actions was intended, the sequence utilized during the thesis will be described along with the justification.

```
READ INPUT
SAVE DATA
ADD USER
DEL USER
ADD TRANS
DEL TRANS
SUBCLUSTER
CLUSTER
COMPARE
SORT USERS
ADD OFFICE
END
```

Fig D-1. Command Menu

Read Input. The READ INPUT command causes the program to read records from the local file DATAFIL, and to create an initial linked list of user and transaction records in the program work area. Absence of DATAFIL or improperly formatted records in the file will cause abnormal termination of the program. The record layout for records in the file is shown in Figure D-2 below. The last record in the file must contain END in columns 1-3. The transaction count in

Starting Column	Length	Contents	Range of Values
1	10	User Identifier	A/N
11	10	User Office	A/N
21	1	Blank	
22	1	Transaction Count	0-9
23	1	Blank	
24	10	Transaction Identifier	A/N
34	10	Transaction Identifier	A/N
.			
.			
104	10	Transaction Identifier	A/N

Fig D-2. DATAFIL Record Layout

column 22 is the count for transactions on that specific DATAFIL record, and does not restrict the number of transaction records that may be linked to a user record. If a user record has thirty transactions linked to it, these transactions will be on successive records in DATAFIL, each with a count entry stating the number of transactions on that specific DATAFIL record for the user. After reading the entire DATAFIL and encountering the END record, the message "COUNT OF INCOMING USERS: XXX" will be displayed and the command menu will be displayed again.

Save Data. The SAVE DATA command creates a new local output file (DATAFIL) containing records as described in Figure D-2. These records are built by extracting information from the records in the program work area existing when the command is issued. As implemented, the transaction count

in column 22 of each record will be between zero and five, to allow display of the records on a terminal screen without encountering wrap-around. Completion of the SAVE DATA operation is indicated by display of the message "OUTPUT USER COUNT: XXX" and redisplay of the command menu.

Add User. The ADD USER command supports creation of new user records linked into the list of user records presently in the work area. Upon input of the command a prompt ("ENTER USER ID OR END") is displayed. The desired user identifier (not more than ten alphanumeric characters) should be input. If END is entered, control will return to the command menu. The user identifier entered is matched to the user identifiers currently in the work area. If a match is found, a message ("USER ALREADY EXISTS") is displayed and the prompt for entry of the user identifier will be repeated. If the user identifier is unique a new user record is created and linked to any others in the work area. Once completed, the prompt for entry of a new user identifier will be displayed to allow continuing creation of user records without redisplay of the command menu.

Del User. The DEL USER command permits selective deletion of user records and their associated transaction records from the current work area. Input of the command will be followed by display of the prompt message "ENTER USER ID OR END." Entry of END will cause control to return to the command menu. If a user identifier is entered the

current work area is searched for a match. If no match is found the message "USER DOES NOT EXIST" is displayed and the prompt for entry of the user identifier is repeated. When a match is found, the user record and all transaction records linked to it are deleted from the work area and the message "USER DELETED- XXXXXXXXXX" is displayed. Completion of the deletion is followed by redisplay of the prompt for entry of the user identifier. This allows continual input of deletions without returning to the main command menu.

Add Trans. The ADD TRANS command allows creation of a new transaction record to be linked to a specific user record. Entry of the command is followed by the prompt "ENTER USER ID OR END". The user identifier should be entered and the work area will be searched for a record match. When the user record is found the prompt "ENTER TRANSACTION ID OR END" will be displayed. The transaction identifier will be checked for duplication in the user record and if not a duplicate, a transaction record is created and linked to the user record. The record creation is followed by display of the prompt for entry of a new transaction identifier if more are to be added. If the transaction entered is already linked to the user record the message "USER ALREADY HAS THIS TRANSACTION" is displayed, and the prompt is repeated. Entry of END for either the transaction or user prompt causes return to the command menu.

Del Trans. The DEL TRANS command allows deletion of a specific transaction record from a user record. In response to the command, the "ENTER USER ID OR END" prompt is displayed to identify the user involved. The input user identifier will be checked for a match to the records in the work area and the prompt "ENTER TRANSACTION ID OR END" will be displayed if a match is found. The transaction identifier entered is used to check all transaction records linked to that user. If a match is found, the transaction record is deleted and the message "TRANSACTION DELETED- XXXXXXXXXX" is displayed and followed by return to the prompt for entry of another transaction identifier. Input of END to either prompt message will cause control to return to the command menu. If no match on the user or the transaction identifier is found, an appropriate message is displayed and the prompt is repeated.

Subcluster. The SUBCLUSTER command is used to reduce the current work area file to only those user records with the office identifier specified in response to the "ENTER OFFICE ID OR END" message. All transaction records linked to the selected user records are also retained for following analysis activities. Input of the SUBCLUSTER command is followed by display of a message warning that use of this command destroys the currently existing work area database. The warning is intended to remind the user that the current work area may need to be saved before proceeding with the

subcluster process. If YES is answered to the warning message, the prompt for entry of the office identifier is displayed. The office identifier is used to search the work area and create a new linkage pattern which includes only those user records with the specific office identifier. Completion of the subcluster process is indicated by return to the command menu display. Since no data edits are applied to the actual data in a user record, the data item OFFICE could contain location information or the building or room number of the user. Performing the subcluster process based on such an attribute could then be followed by use of the COMPARE command to create similarity matrices for the clustered users.

Cluster. The CLUSTER command is similar to the subcluster just described in that it allows the work area database to be reduced in size based on common user attributes. The CLUSTER command allows input of transaction identifiers which are used to determine which user records will be retained in the work area. Entry of the command is followed by display of the prompt "ENTER TRANSACTION ID OR END." When the transaction identifier is entered, only user records with that transaction identifier linked to them will be retained. Once the work area has been reconfigured based on one transaction identifier, the prompt is repeated and a new transaction identifier may be input to be applied to the remaining user records. The concept behind development of

this capability was the possibility of having designed a distributed system node which would support a fixed set of functions or transactions, and desiring to use the similarity matrix process only on those user records requiring all of the available functions. Once two select transaction identifiers have been applied to the database, a list of all users remaining in the work area is displayed. In this way, the user may continue to apply more selection identifiers until only the desired number of user records remain. Input of END to the prompt message causes return of control to the command menu.

Compare. The COMPARE command supports generation of similarity matrices which depict the percentage of transaction identifier commonality between users in the work area database. The computation of the similarity value is accomplished by counting the total number of unique transaction identifiers linked to two user records, adding the number of common transaction identifiers linked to the two, and dividing the number in common by this total count. This comparison process is very time consuming and should be used only on user record groups of some manageable size. For estimation purposes, a user group of 125 users and approximately 2500 linked transaction records required almost one minute of CPU time for generation of the complete matrix. Since an interactive session is limited to two minutes of CPU time on the ASD Cyber, only one such comparison can be accommodated during a

session. The output from the COMPARE process is a local file (MTRIXFIL). The records on the output file are grouped with a user identifier on a line by itself, followed by records with four users and their associated similarity values as compared to that user. The BUILDMATRIX procedure is used to create the matrix file and compute the similarity values. The users to be included in the output file can be controlled based on the amount of similarity that exists between two users. The BUILDMATRIX constant SIMLIMIT is used to compare against the computed similarity value, and if the computed value is greater than SIMLIMIT the user identifier and the computed value will be written to the file. If the computed value is less than or equal to SIMLIMIT no write occurs. Completion of the compare process is indicated by return to the command menu.

Sort Users. The SORT USERS command is used to sort the user records in the work area database into order based on ascending user identifier. This command has no impact on the file DATAFIL, thus the sorted file should be saved upon completion of the command. Completion of the command is indicated by return to the command menu.

Add Office. The ADD OFFICE command permits entry of ten characters of identifying data into the user record data item OFFICE. As mentioned earlier, no editing on the input data is performed so any type of identification data can be placed in this data item to be used as the select parameter

for the SUBCLUSTER command. In response to the ADD OFFICE command the prompt "ENTER OFFICE ID OR END" is displayed. Once the office identifier is entered, the prompt "ENTER USER ID FOR OFFICE, OR END" is displayed. The user identifier for the record to receive this office identifier should be entered. The user record will be searched for, and when found the office identifier will be stored. Completion of this process will be followed by display of the prompt for entry of a user identifier so that multiple user records may receive the same office identifier if so desired. If no match on the user identifier is found, an appropriate message is displayed and the prompt for entry of a user identifier is repeated. Input of END causes return of control to the command menu.

End. The END command causes termination of the interactive program. Prior to input of this command, the user should have saved the work area database if it contains data to be retained.

Summary

This software package has been developed to be as flexible as possible. Changes made to the record data item names for users and transactions should in many instances provide the package user with a clear depiction of the attributes being created and maintained. The modular construction will hopefully allow additional capabilities to be added as further

investigation into distributed processing concepts provide analysis methodologies.

VITA

Ronald Vernon Brandt was born on 13 January 1946 in Saginaw, Michigan. He graduated from Parkside High School in Jackson, Michigan in 1964 and enlisted as a communications operator in the Air Force, after one year of junior college, in 1965. He received the degree of Bachelor of Business Administration from the University of Oklahoma, under the Airman Education and Commissioning Program, and was commissioned in 1973. Before entering the School of Engineering, Air Force Institute of Technology in July 1980, he served on the staff of the Directorate of Personnel Data Systems at the Air Force Manpower and Personnel Center as Chief, Master File Update Control Section, and Chief, APDS-II Requirements Analysis Section. He is a member of the Association for Computing Machinery.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/81D-4	2. GOVT ACCESSION NO. AD-A115 533	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A DISTRIBUTED PROCESSING DESIGN FOR THE PERSONNEL DATA SYSTEM'S CENTRAL SITE		5. TYPE OF REPORT & PERIOD COVERED MS THESIS 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ronald V. Brandt Captain USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Manpower and Personnel Center (MPCD) Randolph AFB, Texas 78150		12. REPORT DATE Dec 1981 13. NUMBER OF PAGES 143
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) 15 APR 1982		
18. SUPPLEMENTARY NOTES Approved for public release; IAW AFR 190-17 ~~FREDERIC C. LYNCH, MAJ~~, USAF ~~Director of Public Affairs~~ Lynn E. Wolaver Dean for Research and Professional Development Air Force Institute of Technology (ATC) Wright-Patterson AFB, OH 45433		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Distributed Processing Clustering		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A Distributed processing system design strategy was developed and applied to the central site portion of the Personnel Data System at the Air Force Manpower and Personnel Center (AFMPC), Randolph Air Force Base, Texas. A software package was developed to support building of a user database and the clustering of users based on geographic locations and similarity of system function use. Design consideration such as data redundancy and concurrency controls are discussed, along with the concept of Conflict		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Graph Analysis developed to support the System for Distributed Databases.

Software documentation and similarity tables depicting the distributed system design involving 425 terminals clustered into 46 user groupings are included. The design as presented requires further development to include data access requirements and definition of lower levels of system function utilization. Development of a simulation package to validate the final system design is recommended.

ILMED

8